Broadview
www.broadview.com.cn

大数据

技术与应用实践指南

Big Data | Technology and Application Practice

赵刚◎著

中国工程院院士 倪光南 倾情作序

雷万云 | 毛新生 | 段永朝 | 安晖 联合力荐

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

大数据

技术与应用实践指南

Big Data Technology and Application Practice

赵刚◎著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

大数据是互联网、移动应用、社交网络和物联网等技术发展的必然趋势，大数据应用成为当前最为热门的信息技术应用领域。本书由浅入深，首先概述性地分析了大数据的发展背景、基本概念，从业务的角度分析了大数据应用的主要业务价值和业务需求，在此基础上介绍大数据的技术架构和关键技术，结合应用实践，详细阐述了传统信息系统与大数据平台的整合策略，大数据应用实践的流程和方法，并介绍了主要的大数据应用产品和解决方案。最后，对大数据面临的挑战和未来的趋势进行了展望。

本书既具有技术深度，又具有很强的可操作性，提供了一个系统性、架构性的大数据应用实践指南，纲要性地指导大数据应用实践，推动大数据技术在各个行业的广泛应用。

图书在版编目（CIP）数据

大数据：技术与应用实践指南 / 赵刚著. —北京：电子工业出版社，2013.10
ISBN 978-7-121-21560-5

Ⅰ. ①大… Ⅱ. ①赵… Ⅲ. ①数据处理－指南 Ⅳ. ①TP274-62

中国版本图书馆 CIP 数据核字（2013）第 228258 号

策划编辑：董　英
责任编辑：付　睿
印　　刷：北京中新伟业印刷有限公司
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　　印张：18.25　　字数：366 千字
印　　次：2013 年 10 月第 1 次印刷
印　　数：4000 册　　定价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

序

随着新一代信息技术的发展和应用，尤其是互联网、物联网、移动互联网、社交网络等技术的发展，我们正在进入一个大数据的时代。从大数据的理念到 Hadoop 开发技术，介绍大数据的书刊纷纷出现，但很多读者看了后可能仍感到不解渴，究其原因是这些书刊没有为读者构建一座连接宏观的理念和深奥的技术细节之间的桥梁，而有关大数据系统性应用实践的书籍则更是凤毛麟角。为此，我向大家推荐这本书，它从大数据技术应用的角度切入，建立了大数据业务价值与技术架构之间的映射关系，内容丰富，条理清晰，深入浅出，繁简适度，使读者能够系统地了解大数据的技术应用体系。

大数据从数据挖掘、商业智能发展而来，是信息技术发展的必然产物。国家“十二五”规划要大力发展包括新一代信息技术在内的战略性新兴产业，大数据就是新一代信息技术的重要领域。它不仅是一次技术领域的革新，因此不仅技术人员必须了解它、研究它、运用它，而且它还将推动企业创新和社会变革，因此各行各业的人员都必须重视它、发展它、推动它。

大数据应用不能一蹴而就，必须遵循科学的方法循序渐进。无论是从业务的角度还是从技术的角度，要将大数据应用讲清楚都不大容易，尤其是要使非本领域的专家能对大数据有一个全面的了解更非易事。为了帮助读者建立起对大数据应用全面、系统的认识，而不只是知道一些零散的技术或服务术语，作者站在系统论的高度对大数据应用做了高度的概括，涵盖大数据的基本概念、业务需求、技术架构、应用集成、实践方法、产业链和制度保障等七个方面，也构成了本书的七个章节。这种结构化、系统化的思想贯穿全书，成为本书的一大特色。这不仅对一般读者，而且对与大数据有关的管理人员和技术人员，都有帮助，使他们可以全面深刻地理解和把握复杂的大数据。

作者提出了大数据应用的业务流程，分析了行业共性业务需求和个性业务需求，并且详细阐述了满足这些业务需求的大数据技术，也介绍了新的大数据技术和现有技术架构的整合。大数据在一些互联网公司有了很好的应用，其他行业也在关注大数据。本书列举出一些实例，给出了大数据应用的流程和方法论，强调了大数据对商业社会的巨大的变革力

量。虽然大数据还是一个新事物，开始时人们难免对其有所怀疑，不敢贸然使用，但越来越多的“吃螃蟹者”已经证明大数据能创造重大的社会效益和经济效益。在当前这场大数据引领的变革浪潮面前，我们应当直面挑战、勇于创新，大胆地应用大数据技术。实际上，在激烈的市场竞争中，不创新的风险往往比创新的风险更大。

本书对大数据的写作高屋建瓴、深入浅出，这与作者的背景是分不开的。赵刚博士一直在中国电子信息产业发展研究院从事信息技术应用研究、咨询和实践工作，承担了多项信息技术战略规划和应用实施项目，有丰富的企业级信息架构的规划和建设经验。2013年，又创办了北京赛智时代信息技术咨询公司，致力于企业级大数据技术的应用咨询和实施工作，发布了银行、保险、电子商务等行业大数据应用研究报告，在大数据应用领域做了很多工作。作者从事产业研究、信息化咨询和信息系统集成的多重背景和学术造诣，使作者能把大数据的业务需求、技术架构和产业链分析在一本书中上下呼应、融会贯通地阐述清晰。

作者在本书最后提出，大数据是中国国内企业迎头赶上的大好机会。我们相信，越来越多的中国大数据公司将会用自己的创新实践证明这一点，中国完全有可能乘大数据的变革之机实现中国信息产业的跨越式发展。

综上所述，本书可以为一切想了解大数据技术应用、建设大数据企业级应用架构、享受大数据分析之美的读者提供一把开启大数据世界的钥匙，即使是对大数据有所研究的人士，本书系统性的视角也可以使他们了解全局、开阔思路，本书具有很高的参考价值。

中国工程院院士　倪光南

前　言

随着互联网、移动互联网、社交网络、物联网、云计算等新一代信息技术的应用和推广，人类产生的数据成倍增长，数据种类繁多，数据在宽带网络中高速流动，数据的待开发价值越来越大，我们已经进入了大数据时代！短短两三年，大数据的理念已经深入人心，大数据的技术也层出不穷，但大数据技术的应用才刚刚开始。本书把阐述的视角放在了大数据的技术应用上，通过分析大数据应用的关键成功因素，希望为政府、行业和企业的大数据技术开发和应用人员提供一本框架性和系统性的技术与应用实践指南。

全书共分为 7 章。

第 1 章是大数据的概念和发展背景，回顾大数据理念和技术的发展历程，梳理大数据发展脉络，并从大数据的体量、数据类型、速度和潜在价值等 4 个特征定义大数据。

大数据的技术应用是为了实现业务的价值，所以第 2 章分析大数据应用的业务需求，梳理企业级大数据应用的业务流程，剖析大数据应用对于组织的业务价值，并深入分析互联网、零售、金融、电信、能源等 9 个行业的大数据应用需求，总结企业级大数据应用的客户分析、绩效分析和风险分析等共性需求。

第 3 章阐述大数据应用的总体架构和关键技术。总体架构分析基于 Apache 开源的大数据平台总体架构参考模型，涵盖了大数据处理、大数据存储、大数据访问、大数据调度、大数据分析展现、大数据与传统数据库连接、大数据管理、安全和备份恢复框架等技术，它能够为企业建设大数据应用平台提供框架参考。基于这一架构，本章进一步详细介绍了大数据存储和处理、大数据查询分析、大数据高级分析和可视化等 3 个方面的关键技术。Hadoop 是大数据技术的内核，本章详细介绍了 Hadoop 三大核心技术，即分布式文件系统 HDFS、分布式计算框架 MapReduce、分布式数据库 HBase 的技术原理、技术构成和应用示例，也介绍了 Hadoop 之外的内存计算、流计算等框架。大数据查询和分析技术介绍了 SQL on Hadoop 技术，包括 Hive、Impala 等技术。大数据高级分析和可视化技术也是大数据的关键技术，本章总体阐述了大数据挖掘与高级分析的算法和技术，对非结构化复杂数据分析、预测分析和开源的 R 语言进行了重点介绍，并介绍了大数据可视化的一些工具。

第 4 章阐述大数据技术应用与企业级应用系统的整合策略。现有企业级数据分析是以关系型数据库为基础的，建立了涵盖网络、存储、服务器、虚拟化、云计算和信息安全等

方面的企业 IT 架构，大数据技术的企业级应用需要实现与这些技术的高效整合，构建新一代的企业级应用架构。本章分别介绍了大数据传输、集成和流程化管理，大数据与存储架构的整合，大数据对网络架构的发展，大数据与虚拟化技术的整合，云计算平台上的大数据云，以及大数据与信息安全等 6 个方面的内容。

第 5 章介绍了大数据企业级应用的实践方法论和应用案例。大数据应用的实践方法论阐述了业务需求定义、现状分析、架构规划和设计、技术切入与实施，以及试用、评估和推广等大数据应用的开发流程。对亚马逊、雅虎、淘宝等互联网企业应用案例的分析，则试图给大数据技术应用实践提供技术细节和实施规模的参考。

第 6 章介绍了大数据应用的主流商业解决方案，首先介绍大数据产业链上的主要厂商，并进一步介绍了 9 家主流厂商的解决方案。

第 7 章是对大数据应用中未来挑战和发展趋势的分析。主要讨论了隐私保护、技术标准、大数据治理等应用发展中的关键挑战和应对策略，最后预测了大数据应用下商业生活的发展趋势。

全书以某商业银行基于大数据的客户分析为案例，便于读者根据案例所阐述的应用场景，结合自身需求学习和掌握大数据技术的应用。

本书的写作最大程度地得益于从事大数据技术研发、应用和研究的社区、业界同仁和爱好者。作者起的作用仅仅是穿针引线，将大数据技术应用开拓者们分享的研究和应用心得总结起来，希望有助于更多技术研发、应用人员和爱好者系统地学习和应用大数据，本书也提供了这些成果的链接，读者可以更加深入地去学习和研究。当然，本书基于作者在信息化领域多年的研究、咨询和系统集成的实践经验，也基于作者所创立的北京赛智时代信息技术咨询有限公司（www.CIOManage.com）在大数据领域的研究成果。本书引用了 CIOManage（赛智时代）的《2013 年中国大数据应用价值研究报告》的很多研究成果。感谢所有为大数据技术应用而努力的同仁们！

本书付梓之际，作者诚惶诚恐，大数据技术远未成熟，大数据技术应用也刚刚拉开帷幕，这样一本技术应用实践指南一定存在诸多问题。但技术应用本来就是一个不断改进和优化的过程，希望我和读者在共同学习和应用的过程，逐步总结出更为精确和实用的经验。欢迎读者与我交流，联系信息如下。

- 微博：http://weibo.com/blogbot
- 博士博客：http://blog.sina.com.cn/blogbot
- 邮箱：blogbot@sina.com

赵刚

2013 年 7 月 29 日于北京嘉铭园

目　　录

第 1 章

大数据的概念和发展背景

本章阐述大数据的概念、发展背景和内涵等。

1.1 大数据的发展背景

在 20 世纪 90 年代后期，当气象学家在做气象地图分析、物理学家在建立大物理仿真模型、生物学家在建立基因图谱的分析过程中，由于数据量巨大，他们已经不能再用传统的计算技术来完成这些任务时，大数据的概念在这些科学研究领域首先被提出来。面对大量科学数据在获取、存储、搜索、共享和分析中遇到的技术难题，一些新的分布式计算技术陆续被研究和开发出来。

2008 年，随着互联网和电子商务的快速发展，当雅虎、谷歌等大型互联网和电子商务公司不能用传统手段解决他们的业务问题时，大数据的理念和技术被他们实际应用。他们遇到的共性问题是，处理的数据量通常很大（那时是 PB 级，1 个 PB 的数据相当于 50% 的全美学术研究图书馆藏书资讯内容），数据的种类很多（文档、日志、博客、视频等），数据的流动速度很快（包括流文件数据、传感器数据和移动设备的数据的快速流动）。而

且，这些数据经常是不完备甚至是不可理解的（需要从预测分析中推演出来）。大数据的新技术和新架构正是在这种背景下被不断开发出来的，以有效地解决这些现实的互联网数据处理问题。

2010 年，全球进入 Web 2.0 时代，Twitter（推特）、Facebook（脸书）、博客、微博、微信等社交网络将人类带入自媒体时代，互联网数据快速激增。随着苹果、三星等智能手机的普及，移动互联网时代也已经到来，移动设备所产生的数据海量般地涌入网络。为了实现更加智能的应用，物联网技术也逐步被推广，随之而来的是更多实时获取的视频、音频、电子标签（RFID）、传感器等数据也被联入互联网，数据量进一步暴增。根据美国市场调查公司 IDC 的预测[1]，人类产生的数据量正在呈指数级增长，大约每两年翻一番，这个速度在 2020 年之前会继续保持下去。全球在 2010 年正式进入 ZB 时代（1 个 ZB 的数据相当于全世界海滩上的沙子数量的总和），预计到 2020 年，全球将总共拥有 35ZB 的数据量。这意味着人类在最近两年产生的数据量相当于之前产生的全部数据量。人类真正进入了一个数据的世界，大数据技术有了用武之地，大数据技术和应用空前繁荣起来。

2011 年，全球著名战略咨询公司麦肯锡的全球研究院（MGI）发布了《大数据：创新、竞争和生产力的下一个新领域》研究报告[2]，这份报告分析了数字数据和文档的爆发式增长的状态，阐述了处理这些数据能够释放出的潜在价值，分析了大数据相关的经济活动和业务价值链。这篇报告在商业界引起极大的关注，为大数据从技术领域进入商业领域吹响了号角。

2012 年 3 月 29 日奥巴马政府以“大数据是一个大生意（Big Data is a Big Deal）”[3]为题发布新闻（如图 1-1 所示），宣布投资 2 亿美元启动“大数据研究和发展计划”，涉及美国国家科学基金、美国国防部等 6 个联邦政府部门，大力推动和改善与大数据相关的收集、组织和分析工具及技术，以推进从大量的、复杂的数据集合中获取知识和洞见的能力。美国政府认为大数据技术事关美国国家安全、科学和研究的步伐。

[1] 2010 年 IDC 提供给 EMC 的报告，见 http://www.emc.com/about/news/press/2010/20100504-01.htm。

[2] http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation。

[3] http://www.whitehouse.gov/blog/2012/03/29/big-data-big-deal。

图 1-1　美国白宫发布的大数据新闻

2012 年 5 月，联合国发布了一份大数据白皮书[4]，总结了各国政府如何利用大数据更好地服务公民，指出大数据对于联合国和各国政府来说是一个历史性的机遇，联合国还探讨了如何利用包括社交网络在内的大数据资源造福人类。

2012 年 12 月“世界经济论坛”发布《大数据，大影响》报告[5]，阐述大数据为国际发展带来的新的商业机会，建议各国与工业界、学术界、非营利性机构与管理者一起利用大数据所创造的机会。

2012 年以来，大数据成为全球投资界最青睐的领域之一，IBM 公司通过并购数据仓库厂商 Netezza、软件厂商 InfoSphere BigInsights 和 Streams 等来增强自己在大数据处理上的实力；EMC 公司陆续收购 Greenplum（Pivotal）、VMWare、Isilo 等公司，展开大数据和云计算产业的战略布局；惠普公司通过并购 3PAR、Autonomy、Vertica 等公司实现了大数据产业链的全覆盖。业界主要的信息技术巨头都纷纷推出大数据产品和服务，力图抢占市场先机。

2012 年以来，国内互联网企业和运营商率先启动大数据技术的研发和应用，如新浪、淘宝、百度、腾讯、中国移动、中国联通、京东商城等企业纷纷启动了大数据试点应用项

[4] http://www.unglobalpulse.org/sites/default/files/BigDataforDevelopment-GlobalPulseMay2012.pdf。

[5] http://www3.weforum.org/docs/WEF_TC_MFS_BigDataBigImpact_Briefing_2012.pdf。

目，推进大数据应用。

2013 年，第 4 期《求是》杂志刊登中国工程院邬贺铨院士的《大数据时代的机遇与挑战》[6]一文，阐述中国科技界对大数据的重视，郭华东、李国杰、倪光南、怀进鹏等院士也纷纷撰文阐述大数据的战略意义。工信部软件服务业司陈伟司长指出：“可以理解，大数据是云计算、物联网、移动互联网、智慧城市等新技术、新模式发展的必然产物”，表明产业主管部门对大数据发展的高度关注。

1.2 大数据的概念和特征

1.2.1 大数据的概念

大数据是指无法在一定时间内用传统数据库软件工具对其内容进行抓取、管理和处理的数据集合（引自维基百科[7]）。

这个定义并不严谨，但这是各种学术和应用领域最广泛引用的一个定义，如果接着以大数据的四个特征作为补充，就能给出一个较为清晰的大数据的概念。

1.2.2 大数据的特征

大数据有四个主要特征。

1. Volume：数据量巨大

体量大是大数据区分于传统数据最显著的特征。一般关系型数据库处理的数据量在 TB 级，大数据所处理的数据量通常在 PB 级以上。

2. Variety：数据类型多

大数据所处理的计算机数据类型早已不是单一的文本形式或者结构化数据库中的表，

[6] 邬贺铨，《大数据时代的机遇与挑战》，《求是》杂志，2013 年 2 月。

[7] http://en.wikipedia.org/wiki/Big_data。

它包括订单、日志、BLOG、微博、音频、视频等各种复杂结构的数据。

3. Velocity：数据流动快

速度是大数据区分于传统数据的重要特征。在海量数据面前，需要实时分析获取需要的信息，处理数据的效率就是组织的生命。

4. Value：数据潜在价值大

在研究和技术开发领域，上述三个特征已经足够表征大数据的特点。但在商业应用领域，第四个特征就显得非常关键！投入如此巨大的研究和技术开发的努力，就是因为大家都洞察到了大数据的潜在的巨大价值。如何通过强大的机器学习和高级分析更迅速地完成数据的价值“提纯”，挖掘出大数据的潜在价值，这是目前大数据应用背景下亟待解决的难题。

1.3　大数据的产生

大量数据的产生是计算机和网络通信技术（ICT）广泛应用的必然结果，特别是互联网、云计算、移动互联网、物联网、社交网络等新一代信息技术的发展，起到了催化剂的作用，它带来了数据产生的四大变化：一是数据产生由企业内部向企业外部扩展；二是数据产生由 Web 1.0 向 Web 2.0 扩展；三是数据产生由互联网向移动互联网扩展；四是数据产生由计算机/互联网（IT）向物联网（IOT）扩展。这 4 个变化，让数据产生源头成倍地增长，数据量也大幅度地快速增长。

1.3.1　数据产生由企业内部向企业外部扩展

在企业内部的企业资源计划（ERP）、办公自动化（OA）等业务、管理和决策分析系统所产生的数据，主要存储在关系型数据库中。内部数据是企业内最成熟并且被熟知的数据。这些数据已经通过多年的 ERP、主数据管理（MDM）、数据仓库（DW）、商业智能（BI）和其他相关应用积累，实现了内部数据的收集、集成、结构化和标准化处理，可以为企业决策提供分析报表和商业智能。

一些企业已经关注到内部交易数据的潜在价值，如利用一些非结构化数据的分析方法，挖掘在客户交易过程、业务处理流程和电子邮件中所获得的内部日志等数据，来为企业提供客户分析、绩效和文化等方面的更多洞察力。还有一些企业内部的数据量也很大，如电信运营商、石油勘探企业等，这些机构使用大数据有多年时间了。例如，一家全球电信公司每天从 120 个不同系统中收集数十亿条详细呼叫记录，并保存至少 9 个月时间；一家石油勘探公司分析几万亿字节的地质数据。对于这些公司，大数据虽然是一个新概念，但要做的事情却并不新鲜。他们早就在使用大数据，但由于没有合适的技术手段对这些大数据进行分析，这些大数据大部分被丢弃了。

对于所有企业而言，信息化的应用环境在发生着变化，外部数据迅速扩展。企业应用和互联网应用、移动互联网应用的融合越来越快，企业需要通过互联网来服务客户、联系外部供应商、沟通上下游的合作伙伴，并在互联网上实现电子商务和电子采购的交易。企业需要开通微博、博客等社交网络来进行网络化营销、客户关怀和品牌建设。企业的产品被贴上了电子标签，在制造、供应链和物流的全程中进行跟踪和反馈。伴随着自带设备（BYOD）工作模式的兴起，企业员工自带设备进行工作，个人的数据进一步与企业数据相融合，必将产生更多来自企业外部的数据。

企业内外部数据的产生如表 1-1 所示。

表 1-1　企业内外部数据的产生

	企业内部数据	企业外部数据
企业应用	ERP、CRM、MES、SCADA、OA、专业业务系统、传感器	电子商务、电子采购、知识管理 呼叫中心、企业微博、企业微信 RFID、传感器、BYOD
数据规模	TB 级	PB 级
数据存储	关系型数据库、数据仓库	各种格式的文档

来源：CIOManage（赛智时代）

1.3.2　数据产生从 Web 1.0 向 Web 2.0、从互联网向移动互联网扩展

随着社交网络的发展，互联网进入了 Web 2.0 时代，每个人从数据的使用者，变成了数据的生产者，数据规模迅速扩张，每时每刻都在产生大量的新数据。例如，从全球统计数据来看，全球每秒钟发送 290 万封电子邮件，每秒钟电子商务公司亚马逊上将产生 72.9

笔商品订单，每分钟会有 20 个小时的视频上传到视频分享网站 YouTube，谷歌上每天需要处理 24PB 的数据，Twitter 上每天发布 5 千万条消息，每天被每个家庭消费的数据有 375MB，每个月网民在 Facebook 上要花费 7 千亿分钟……

从中国来看，数据规模也十分巨大，淘宝网目前已拥有近 5 亿的注册会员，在线商品 8.8 亿，每天交易超过数千万笔，其单日数据产生量超过 20TB。百度目前数据总量接近 1000PB，存储网页数量接近 1 万亿，每天大约要处理 60 亿次搜索请求，几十 PB 数据。新浪微博每天有数十亿外部网页和 API 接口访问需求，服务器群在晚上高峰期每秒要接受 100 万个以上的响应请求。

移动互联网的发展让更多人成为数据的生产者，据统计全球每个月移动互联网使用者发送和接收的数据高达 1.3EB。在中国，中国联通用户上网记录条数为 83 万条/秒，即一万亿条/月，对应数据量为 300TB/月，或 3.6PB/年。

1.3.3　数据产生从计算机/互联网（IT）向物联网（IOT）扩展

随着视频、传感器、智能设备和 RFID 等技术的增长，视频、音频、RFID、机器对机器（M2M）、物联网和传感器等数据大量产生，其数据量更是巨大。根据 IDC 公布的数据，2005 年仅由 M2M 产生的数据占全世界数据总量的 11%，预计到 2020 年这一数值将增加到 42%。思科（Cisco）公司预测，仅仅移动设备的数据流量将在 2015 年达到每月 6.3 EB 的规模。

1.4　数据的量级

1.4.1　数据大小的量级

数据量的大小是用计算机存储容量的单位来计算的，基本的单位是字节（Byte），每一级按照千分位递进，如下所示：

```
1Byte(B)                相当于一个英文字母
1Kilobyte(KB)=1024B     相当于一则短篇故事的内容
1Megabyte(MB)=1024KB    相当于一则短篇小说的文字内容
1Gigabyte(GB)=1024MB    相当于贝多芬第五乐章交响曲的乐谱内容
```

1Terabyte(TB)=1024GB	相当于一家大型医院中所有的 X 光图片内容
1Petabyte(PB)=1024TB	相当于 50%的全美学术研究图书馆藏书信息内容
1Exabyte (EB)=1024PB	5EB 相当于至今全世界人类所讲过的话语
1Zettabyte(ZB)=1024EB	如同全世界海滩上的沙子数量的总和
1Yottabyte(YB)=1024ZB	1024 个像地球一样的星球上的沙子数量的总和

1.4.2 大数据的量级

目前，传统企业的数据量基本在 TB 级以上，一些大型企业达到了 PB 级，谷歌、百度、新浪、腾讯、淘宝这些企业的数据量在 PB 级以上。

大数据技术和应用擅长处理的数量级一般都在 PB 级以上。但数据量的巨大是相对处理这些数据的计算设备而言的，例如，对一台小型机或 PC 服务器，PB 级数据是大数据，但可能对一台智能手机而言，GB 级的数据就是“大数据”。就目前大数据技术架构所处理的数据来看，通常是指 PB 级以上的数据。

摩尔定律是由英特尔（Intel）创始人之一戈登·摩尔（Gordon Moore）提出来的，其内容为：当价格不变时，集成电路上可容纳的晶体管数目约每隔 18 个月便会增加一倍，性能也将提升一倍。这一定律揭示了信息技术进步的速度。吉姆·格雷（Jim Gray）的新摩尔定理认为，每 18 个月全球新增的信息量是计算机有史以来全部信息量的总和，数据容量每 18 个月就翻一番。据 IDC 统计，全球在 2010 年正式进入 ZB 时代，预计到 2020 年，全球将总共拥有 35ZB 的数据量。但是，过去的 50 年，数据存储的成本大概每两年就能降一半，而存储密度却增加了 5000 万倍。

因此，我们的世界正在成为一个数据的世界，我们正处于大数据时代的边缘，像水、空气、石油一样，数据正成为这个世界中的一种自然资源。

1.5 大量不同的数据类型

大数据不仅仅体现在数量大，也体现在数据类型多。如此海量的数据中，仅有 20%左右属于结构化的数据，80%的数据属于广泛存在于社交网络、物联网、电子商务等领域的非结构化数据。由我们创造的技术产生的这些数据早已经远远超越了目前人力所能处理的范畴，机器数据日益重要，且数据越来越成为一种自然资源。

1.5.1　按照数据结构分类

按照数据结构，数据分为结构化数据、半结构化的非结构化数据和无结构的非结构化数据。结构化数据是存储在数据库里、可以用二维表结构来逻辑表达实现的数据。相对于结构化数据而言，不方便用数据库二维表结构来表现的数据即称为非结构化数据，包括所有格式的办公文档、文本、图片、XML、HTML、各类报表、图像、音频、视频信息等。非结构化数据中又包含半结构化数据和无结构的非结构化数据。

1. 结构化数据

结构化数据的特点是任何一列的数据不可以再细分，任何一列的数据都有相同的数据类型。所有关系型数据库（如 Oracle、SQL Server、DB2、MySQL 等）中的数据全部为结构化数据。关系型数据库存储的结构化数据示例如表 1-2 所示。

表 1-2　结构化数据示例

客户号	客户姓名	交易额	所购产品
200048901	张伟	1000.0	冰箱
200057903	李东	456.0	烤炉

2. 半结构化数据

半结构化数据，是介于完全结构化数据和完全无结构的数据之间的数据，半结构化数据的格式较为规范，一般都是纯文本数据，可以通过某种方式解析得到每项的数据。最常见的就是日志数据、XML、JSON 等格式的数据，它们每条记录可能会有预定义的规范，但是每条记录包含的信息可能不尽相同，也可能会有不同的字段数，包含不同的字段名或字段类型，或者包含着嵌套的格式。这类数据一般都以纯文本的形式输出，管理维护也较为方便，但在需要使用这些数据时，如获取、查询或分析数据时，可能需要先对这些数据格式进行相应的解析。

（1）XML 文档

一个 XML 文档示例如下：

```
<?xml version="1.0"?>
<Order>
<Product xmlns="http://market">
```

```
            <Title>The Joshua Tree</Title>
            <Artist>U2</Artist>
            </Product>
</Order>
```

（2）JSON

JSON（JavaScript Object Notation）是一种基于 JavaScript 的轻量级的数据交换格式，它的格式以键值对（Key/Value）的形式输出数据，示例如下：

```
{ "people": [
{ "firstName": "Brett", "lastName":"McLaughlin", "email": "aaaa" },
{ "firstName": "Jason", "lastName":"Hunter", "email": "bbbb"},
{ "firstName": "Elliotte", "lastName":"Harold", "email": "cccc" }
]}
```

（3）日志文件

日志文件是在计算机系统运行中由计算机或传感器等生成的数据，用于记录业务或信息系统内执行的自动功能的详细信息。最常见的就是 Web 日志，它根据预定义的字段顺序打出相应的值，一个 Web 日志文件的示例如下：

```
2005-01-0316:44:57218.17.90.60GET/Default.aspx-80 -218.17.90.60Mozilla/4
.0+(compatible;+MSIE+6.0;+Windows+NT+5.2;+.NET+CLR+1.1.4322)20000
```

（4）点击流（Click-stream）

客户对企业网站的每一次点击都会被企业网络服务器记录在日志中，由此产生了**点击流**数据，也是日志的一种。

3. 无结构的非结构化数据

无结构的非结构化数据指的是那些非纯文本类数据，没有标准格式，无法直接解析出相应的值。常见的非结构化数据有富文本文档、网页、多媒体（图像、声音、视频等）。这类数据不易收集管理，也无法直接查询和分析，所以对这类数据需要使用一些不同的处理方式。如图 1-2 所示为现实生活中的非结构化数据。

- Web 网页
- 电子邮件
- 富文本文档（Rich Text Format，简称为 RTF）

◎　富媒体文件（Rich Media）

它是具有动画、声音、视频和/或交互性的信息传播媒介，包含下列常见的形式之一或者几种的组合：流媒体、声音、Flash 以及 Java、Javascript、DHTML 等程序设计语言。富媒体可应用于各种网络服务中，如网站、电子邮件、旗帜广告（Banner）、按钮式广告（Button）、弹出式广告、插播式广告等。其中：流媒体文件（Stream Media）是采用流式传输的方式在 Internet/Intranet 播放的媒体格式，如声音流、视频流、文本流、图像流、动画流等。

图 1-2　现实世界中的非结构化数据

◎　实时多媒体数据

它是在各个行业数字化中产生的大量实时多媒体数据，包括各种视频、图像和音频文件，如 CAD/CAM 数据、视频会议、视频监控、数字电影、卫星图像、遥感图像、大型实景游戏、扫描仪数据、医学影像、传感器数据、分享视频、分享照片、数字电视等。

◎　即时消息或事件数据

如 Twitter、微博、微信等。

◎　图数据（含社交网络）

◎　语义 Web（RDF）

1.5.2 按照产生主体分类

1. 最里层：少量企业应用产生的数据

- 关系型数据库中的数据
- 数据仓库中的数据

2. 次外层：大量人产生的数据

- Twitter，每天 5000 万 tweets、每年 1400%的增长率
- 微博（文字、图片和视频）
- 微信（文字、音频、视频）
- 博客、评论、图片和视频分享
- 企业博客、企业微博、企业微信
- 工程师的 CAD/CAM 数据、设计文档、笔记、日志
- 电子商务在线交易的日志数据、供应商交易的日志数据
- 呼叫中心的评论、留言或者电话投诉等
- 企业应用相关评论数据

3. 最外层：巨量机器产生的数据

- 应用服务器日志（Web 站点、游戏）
- 传感器数据（天气、水、智能电网）
- 图像和视频（车间监控的视频数据、交通、安全摄像头）
- RFID、二维码或者条形码扫描的数据

大数据应用需要整合来自不同数据源、采用不同格式、跨不同业务的各类数据。例如，在一个制造企业中，产品创新的创意可能来自电子商务网站的交易数据和社交网站上关于产品的微博评论和转发信息，产品的设计可能需要调用产品知识库中的二维和三维 CAD 设计文档以及三维动画原型，产品的市场宣传可能需要研究竞争产品的视频短片等。而一家医疗机构，需要分析与患者症状相似的很多病人的电子健康档案和电子病历，需要查阅护士和医生的各种病历记录，需要分析患者自服务设备的日志数据以了解近期就诊趋势，需要通过远程的家庭医疗设备分析流媒体数据，这些数据种类也是很多样的。分析大数据的前提是能够采集、解析、集成和存储这些不同类型的数据。对于大量非结构化数据，传

统的关系型数据库明显力不从心。

如图 1-3 所示为不同的大数据主体。

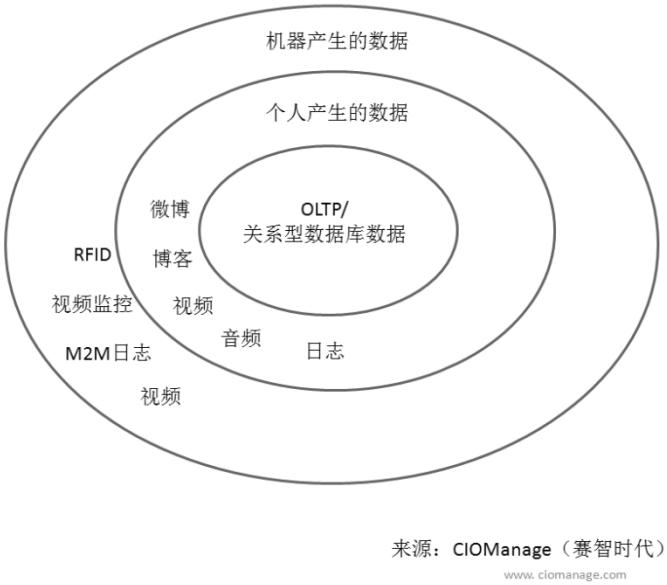


图 1-3　不同的大数据主体

1.5.3　按照数据作用方式分类

按照数据作用的方式，分为交易数据和交互数据。

交易数据是指来自电子商务和企业应用的数据，包括 ERP、企业对企业（B2B）、企业对个人（B2C）、个人对个人（C2C）、团购等系统，这些数据存储在关系型数据库和数据仓库中，可以执行联机事务处理（OLTP）和联机分析处理（OLAP）。这些数据的规模和复杂性一直在提高。

交互数据指来自相互作用的社交网络的数据，包括社交媒体交互（人为生成交互）和机器交互（设备生成交互）的新型数据。

两类数据的有效融合将是大势所趋，大数据应用要有效集成这两类数据，并在此基础上，实现这些数据的处理和分析。

1.6 大数据的速度

大数据的速度是指数据创建、处理和分析的速度，它是由数据从客户端采集、装载并流动到处理器和存储设备、在处理器和存储设备中进行计算的速度所决定的。

在当前的计算环境下，由于处理器和存储等计算技术的不断进步，数据处理的速度越来越快，传统计算技术渐渐不能满足大容量和多种类的大数据的处理速度的要求。在交互式的计算环境下，海量数据被实时创建，用户需要实时的信息反馈和数据分析，并将这些数据结合到自身高效的业务流程和敏捷的决策过程中。大数据技术必须解决大容量、多种类数据高速地产生、获取、存储和分析中间的问题。

一方面要解决大数据容量下的数据时延问题。所谓数据时延是指，从数据创建或获取到数据可以访问之间的时间差。大数据处理需要解决大容量数据处理的高时延问题，需要采用低时延的技术来进行处理。如对一次 PB 级大数据的复杂查询，传统结构化查询语言（SQL）技术可能需要几个小时，基于大数据技术平台希望将这一时延逐步降低到分钟级、秒级、毫秒级、完全实时，大数据技术正在做到这一点。

另一方面要解决时间敏感的流程中实时数据的高速处理问题。对于对时间敏感的流程，例如实时监控、实时欺诈监测或多渠道“实时”营销，某些类型的数据必须进行实时分析，以对业务产生价值，这涉及从数据的批处理、近线处理到在线实时流处理的演变。

1.7 大数据的潜在价值

大数据的价值是与大数据的容量和种类密切相关的。一般来看，数据容量越大，种类越多，信息量越大，获得的知识越多，能够发挥的潜在价值也越大。但这依赖于大数据处理的手段和工具，否则由于信息和知识密度低，可能造成数据垃圾和信息过剩，失去数据的利用价值。

研究表明，数据的价值会随着时间的流逝而降低[8]。简单地看，数据的价值与时间是成反比的。因此，数据处理速度越快，数据价值越能够更好地获得。大数据的价值也与它

[8] 《大数据时代：生活、工作与思维的大变革》，维克托·迈尔·舍恩伯格、肯尼思·库克耶著，浙江人民出版社，2013 年 1 月。

所传播和共享的范围相关，使用大数据的用户越多，范围越广，信息的价值就越大。大数据价值的充分发挥，依赖于大数据的分析和挖掘技术，更好的分析工具和算法能够获得更为准确的信息，也更能发挥其价值。总之，大数据的价值，可以用如下的公式来简单定义：

$$\text{大数据价值 } V=\frac{\text{大数据处理和分析算法和工具 } f(\text{大数据量 } v_1,\ \text{大数据种类 } v_2,\ \text{高速流动 } v_3)\times\text{大数据用户数}}{\text{大数据存在时间 } t}$$

因此，大数据处理和分析的技术对于挖掘大数据价值的作用十分关键。

1.8　大数据的挑战

1.8.1　业务视角不同带来的挑战

以往，企业通过内部 ERP、客户关系管理（CRM）、供应链管理（SCM）、BI 等信息系统建设，建立高效的企业内部统计报表、仪表盘等决策分析工具，为企业业务敏捷决策发挥了很大作用。但是，这些数据分析只是冰山一角，这些报表和仪表盘其实是“残缺”的，更多潜在的有价值的信息被企业束之高阁。大数据时代，企业业务部门必须改变他们看数据的视角，更加重视和利用以往被放弃的交易日志、客户反馈、社交网络等数据。这种转变需要一个接受过程，但实现转变的企业则已经从中获得巨大收益。据有关统计，电子商务企业亚马逊近三分之一的收入来自基于大数据相似度分析的推荐系统的贡献。花旗银行新产品创新的创意很大程度来自各个渠道收集到的客户反馈数据。因此，在大数据时代，业务部门需要以新的视角来面对大数据，接受和利用好大数据，创造更大的业务价值。

1.8.2　技术架构不同带来的挑战

传统的关系型数据库（RDBMS）和结构化查询语言（SQL）面对大数据已经力不从心，更高性价比的数据计算与存储技术和工具不断涌现。对于已经熟练掌握和使用传统技术的企业信息技术人员来说，学习、接受和掌握它需要一个过程，从内心也会认为现在的技术和工具足够好，对新技术产生一种排斥的心理，怀疑它只是一个新的噱头。新技术本身的不成熟性、复杂性和用户不友好性也会加深这种印象。但大数据时代的技术变革已经不可逆转，企业必须积极迎接这种挑战，以包容的方式迎接新技术，以集成的方式实现新老系统的整合。

1.8.3 管理策略不同带来的挑战

大容量和多种类的大数据处理将带来企业信息基础设施的巨大变革，也会带来企业信息技术管理、服务、投资和信息安全治理等方面的新的挑战。如何利用公有云服务来实现企业外部数据的处理和分析？对大数据架构采取什么样的管理和投资模式？对大数据可能涉及的数据隐私如何进行保护？……这些都是企业应用大数据需要面对的挑战。

挑战与机遇并存，但机遇远远大于挑战，大数据应用的热潮已经来到，本书力图指导读者一步一步开启大数据的应用。

第 2 章

大数据应用的业务需求

本章阐述大数据应用的主要应用领域、业务价值和业务需求。

2.1 大数据应用的业务流程

数据处理的流程包括产生数据，收集、存储和管理数据，分析数据，利用数据等阶段。大数据应用的业务流程也是一样的，包括产生数据、聚集数据、分析数据和利用数据 4 个阶段，只是这一业务流程是在大数据平台和系统上执行的，如图 2-1 所示为大数据应用的业务链。

2.1.1 产生数据

在组织经营、管理和服务的业务流程运行中，企业内部业务和管理信息系统产生了大量存储于数据库中的数据，这些数据库对应着每一个应用系统且相互独立，如 ERP 数据库、财务数据库、CRM 数据库、人力资源数据库等。在企业内部的信息化应用中，也产生了非结构化文档、交易日志、网页日志、视频监控文件、各种传感器数据等非结构化数

据，这是在大数据应用中可以被发现潜在价值的企业内部数据。企业建立的外部电子商务交易平台、电子采购平台、客户服务系统等帮助企业产生了大量外部的结构化数据。企业的外部门户、移动 APP 应用、企业博客、企业微博、企业视频分享、外部传感器等系统帮助企业产生了大量外部的非结构化数据。

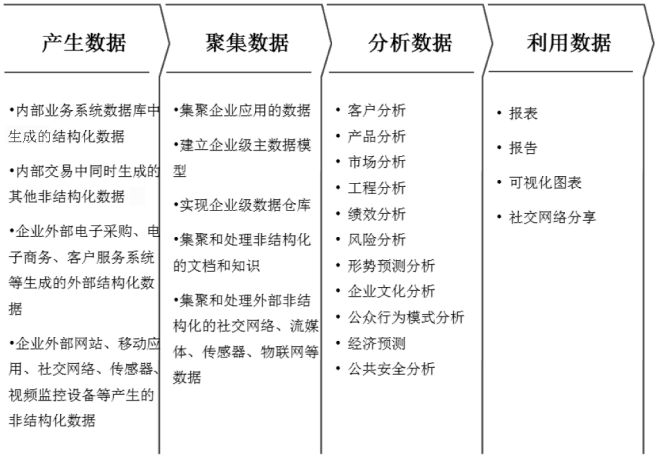


图 2-1　大数据应用的业务链

2.1.2　聚集数据

企业架构（EA）的 3 个核心要素是业务、应用和数据，业务架构描述业务流程和功能结构，应用架构描述处理工具的结构，数据架构描述企业核心的数据内容的组织。企业内外部已经产生了大量的结构化、非结构化数据，需要将这些数据组织和聚集起来，建立企业级的数据架构，有组织地对数据进行采集、存储和管理。首先实现的是不同应用数据库之间的整合，这需要建立企业级的统一数据模型，实现企业主数据管理。所谓主数据是指企业的产品、客户、人员、组织、资金、资产等关键数据，通过这些主数据的属性以及它们之间的相互关系能够建立企业级数据架构和模型。在统一模型的基础上，利用提取、转换和加载（ETL）技术，将不同应用数据库中的数据聚集到企业级的数据仓库（DW），实现企业内部结构化数据的集成，这为企业商业智能分析奠定了一个很好的基础。面对企业内外部的非结构化数据，借助数据库和数据仓库的聚集，效果并不好。文档管理和知识管理是对非结构化文档进行处理的一个阶段，仅限于对文档层面的保存、归类和基于元数

据的管理。更多非结构化文档的集聚，需要引入新的大数据的平台和技术，如分布式文件系统、分布式计算框架、非 SQL 数据、流计算技术等，通过这些技术来加强非结构数据的处理和集聚。内外部结构化、非结构化数据的统一集成则需要实现两种数据（结构化、非结构化）、两种技术平台（关系型数据库、大数据平台）的进一步整合。

2.1.3　分析数据

集成起来的企业各种数据是大容量、多种类的大数据，分析数据是提取信息、发现知识、预测未来的关键步骤。分析只是手段，并不是目的。企业内外部数据分析的目的是为了发现数据所反映的组织业务运行的规律，是为了创造业务价值。对于企业来说，可能基于这些数据进行客户行为分析、产品需求分析、市场营销效果分析、品牌满意度分析、工程可靠性分析、企业业务绩效分析、企业全面风险分析、企业文化归属度分析等。对于政府和其他事业机构，可以进行公众行为模式分析、经济预测分析、公共安全风险分析等。很多数据分析需要借助模型和算法，但一些分析就只是数据驱动的相关性统计。

2.1.4　利用数据

数据分析的结果，不是仅仅呈现给专业做数据分析的数据科学家，而是需要呈现给更多非专业人员才能真正发挥它的价值，客户、业务人员、高管、股东、社会公众、合作伙伴、媒体、政府监管机构等都是大数据使用者。因此，大数据分析结果应当以不同专业角色、不同地位人员对数据表现的不同需求提供给他们，或许是上报的报表、提交的报告、可视化的图表、详细的可视化分析或者简单的微博信息、视频信息。只有数据被重复利用的次数越多，它所能发挥的价值就越大。

2.2　大数据应用的业务价值

维克托 •迈尔 •舍恩伯格认为[1]大数据的重要价值在于建立数据驱动的关于大数据相关关系的分析，而建立在相关关系分析法基础上的预测是大数据的核心！大数据让我们知道“是什么”，也许我们还不明白为什么，但对瞬息万变的商业世界来说，知道是什么比知道

[1] 维克托・迈尔・舍恩伯格，著有《大数据时代——生活、工作与思维的大变革》一书，由浙江人民出版社出版。

为什么更为重要。大数据应用真正要实现的是“用数据说话”，而不是直觉或经验。总结起来，大数据应用的业务价值在于 3 个方面：一是发现过去没有发现的数据潜在价值，二是通过不同数据集的整合创造新的数据价值，三是把在一个领域已经发挥过价值的数据再次应用在新的领域创造出新价值。如图 2-2 所示为大数据对企业的潜在价值。

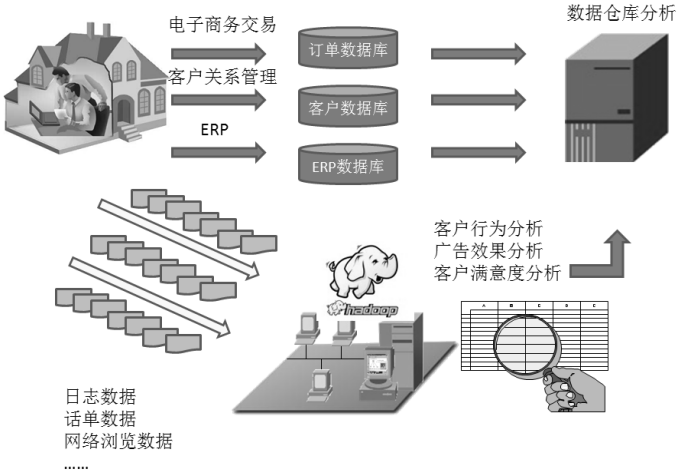


图 2-2　大数据对企业的潜在价值

2.2.1　发现大数据的潜在价值

在大数据应用的背景下，企业开始关注过去不重视、丢弃或者无能力处理的数据，从中分析潜在的信息和知识，用于以客户为中心的客户拓展、市场营销等。例如，企业在进行新客户开发、新订单交易和新产品研发的过程中，产生了很多用户浏览的日志、呼叫中心的投诉和反馈，这些数据过去一直被企业所忽视，通过大数据的分析和利用，这些数据能够为企业的客户关怀、产业创新和市场策略提供非常有价值的信息。

2.2.2　实现大数据整合创新的价值

在互联网和移动互联网时代，企业收集了来自网站、电子商务、客户记分卡、移动应用、呼叫中心、企业微博等不同渠道的客户访问、交易和反馈数据，把这些数据整合起来，形成关于客户的全方位信息，这将有助于企业给客户提供更有针对性、更贴心的产品和服务。

如图 2-3 所示为某运营商的多渠道数据整合分析图。

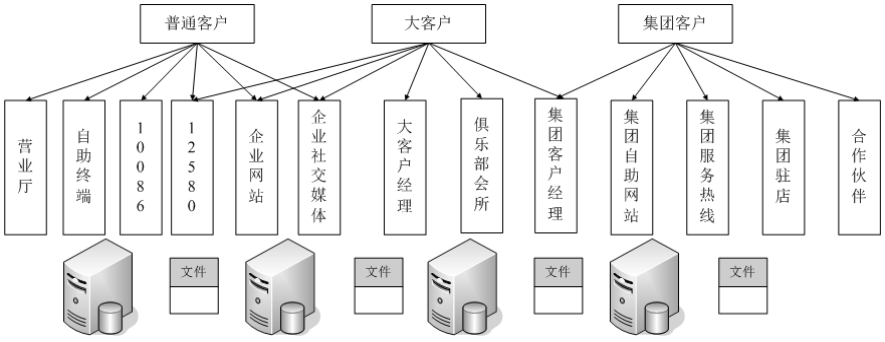


图 2-3　某运营商的多渠道数据整合分析图

2.2.3　新领域再利用的价值

数据是企业的宝贵资源，特别是客户数据、行业数据等。当企业把这些数据从一个业务领域向另外一个业务领域进行再利用，发挥了数据低成本复制和增值的价值。很多成功的互联网企业就是基于原始用户群的数据再利用，不断进行业务创新，在新的领域发挥现有数据的价值。

2.3　各行业大数据应用的个性需求

大数据已成为全球商业界一项优先级很高的战略任务，因为它能够对全球整合经济时代的商务产生深远的影响。大数据在各行各业都有应用，尤其是在公共服务领域具有广阔的应用前景，如政府、金融、零售、医疗等行业。

2.3.1　互联网与电子商务行业

互联网和电子商务领域是大数据应用的主要领域，主要需求是互联网访问用户分析和用户行为分析，基于这些进而实现推荐系统、广告追踪、点击流分析等应用。

1. 用户分析

在 Web 2.0 和电子商务时代，互联网、移动互联网和电子商务上的用户，大部分是注

册用户，通过简单的注册，用户拥有了自己的账户，互联网企业则拥有了用户的基本资料信息，互联网最基本的是用户名/E-mail、密码、性别、年龄、移动电话等基本信息，社交媒体的用户信息包括更多，如新浪微博中用户可以填写自己的昵称、头像、真实姓名、所在地、性别、生日、博客地址、E-mail、QQ/MSN、自我介绍、用户标签、教育信息、职业信息等信息，腾讯 QQ 客户端上可以填写头像、昵称、个性签名、姓名、性别、英文名、生日、血型、生肖、故乡、所在地、邮编、电话、学历、职业、语言、手机等。移动互联网用户的信息与手机绑定，可以获得手机号、手机通信录等用户信息。由于互联网用户在上网期间会留下更多的个人信息，如博客中记录关于妻子、儿女等信息，微博中记录关于个人爱好等信息，校友录中记录关于小学、中学、大学同学等信息，在互联网企业的用户信息库中的用户信息会越来越完整。

2. 用户行为分析

用户访问行为的分析是互联网和电子商务领域大数据应用的重点。用户行为分析可以从行为载体和行为的效果两个维度进行分类[2]。从用户行为的产生方式和载体来分析用户行为主要包括如下几点。

鼠标点击和移动行为分析。在移动互联网之前，互联网上最多的用户行为基本都是通过鼠标来完成的，分析鼠标点击和移动轨迹是用户行为分析的重要部分。目前国内外很多大公司都有自己的系统，用于记录和统计不同程度上的用户鼠标行为。此外，据了解，目前国内的很多第三方统计网站也可以为中小网站和企业提供鼠标移动轨迹等记录。

移动终端的触摸和点击行为。随着新兴的多点触控技术在智能手机上的广泛应用，触摸和点击行为能够产生更加复杂的用户行为，有必要对此类行为进行记录和分析。

键盘等其他设备的输入行为。此类设备主要是为了满足不能通过简单点击等进行输入的场景，如大量内容输入。键盘的输入行为不是用户行为分析的重点，但键盘产生的内容却是大数据应用中内容分析的重点。

眼球，眼动行为。基于此种用户行为的分析在国外比较流行，目前在国内的很多领域也有类似用户研究的应用，通过研究用户的眼球移动和停留等，产品设计师可以更容易了解界面上哪些元素更受用户关注，哪些元素设计得合理或不合理等。

[2] http://didu.blog.techweb.com.cn/archives/112.html。

基于以上这 4 类媒介，用户在不同的产品上可以产生千奇百怪、形形色色的行为，可以通过对这些行为的数据记录和分析更好地指导产品开发和用户体验。

针对不同业务类型的网站，用户行为效果的分析有所不同，如表 2-1 所示为互联网用户行为效果分析。

表 2-1　互联网用户行为效果分析

	消　费　类	贡　献　类
传统内容类网站	页面内元素点击行为 浏览行为与路径 相关推荐内容点击喜好 页面停留时间 浏览器/上网场景特性行为……	点击率排行 回头率……
电子商务网站	网站本身的行为分析 商品浏览选择过程 付费与购买行为 退换货行为……	网站本身的行为分析 收藏 点评 分享……
社交网络	网站本身的行为分析 消费内容的路径更加多样化 ……	网站本身的行为分析 内容发布与创建 内容转发与参与 内容评论 关系链的创建与形成 应用和其他使用习惯……
游戏	游戏操作行为 付费与购买行为 装备的使用喜好……	玩家行为产出的游戏战略和产出物 玩家行为 ……

通过对这些互联网行为数据进行不同方法的建模和推导分析，就可以得出有价值的数据结果，这是互联网和电子商务大数据应用的真正需求。

3. 基于大数据相关性分析的推荐系统

建立推荐系统是互联网和电子商务企业进行大数据应用的一个关键需求。这已经在电子商务企业广泛应用，亚马逊、当当网等电子商务企业就根据大量的用户交易行为数据的相关性分析为读者推荐相关书目，例如，根据同样的兴趣爱好者的付费购买行为，为用户

推荐书目，以同理心来刺激购物消费，图 2-4 是基于大数据的推荐系统示例。有关数据显示，亚马逊、当当网等电子商务企业近 1/3 的收入来自于它的个性化推荐系统。

图 2-4　基于大数据的推荐系统

推荐系统的基础是用户购买行为数据，处理数据的基本算法在学术领域被称为“客户队列群体的发现”，队列群体在逻辑和图形上用链接表示，队列群体的分析很多都涉及特

殊的链接分析算法。推荐系统分析的维度是多样的，例如可以根据客户的购物喜好为其推荐相关书目，也可以根据社交网络进行推荐。如果利用传统的分析方法，需要先选取客户样本，把客户与其他客户进行对比，找到相似性，但是推荐系统的准确性较低。采取大数据分析技术后，大大提高了分析的准确性。

4. 广告追踪和优化

电子商务网站一般都记录包括每次用户会话中每个页面事件的海量数据。这样就可以在很短的时间内完成一次广告位置、颜色、大小、用词和其他特征的试验。当试验表明广告中的这种特征更改促成了更好的点击行为，这个更改和优化就可以实时实施。

5. 内容针对性投放

从用户的行为分析中，可以获得用户偏好，如通过微博用户分析，获悉用户在每天的 4 个时间点最为活跃：早起去上班的路上、午饭时间、晚饭时间、睡觉前。掌握了这些用户行为，企业就可以在对应的时间段做某些针对性的内容投放和推广等。

6. 产品分析

电子商务网站，通过对用户的消费行为和贡献行为产生的数据进行分析，可以量化很多指标服务于产品各个生产和营销环节，如转化率、客单价、购买频率、平均毛利率、用户满意度等指标，从而为产品客户群定位或市场细分提供科学依据。

7. 病毒式（Virality）传播分析

病毒式营销是互联网上的用户口碑传播，这种传播通过社交网络像病毒一样迅速蔓延传播，使得它成为一种高效的信息传播方式。对于病毒式营销的效果分析是非常重要的，不仅可以及时掌握营销信息传播所带来的反应（例如对于网站访问量的增长），也可以从中发现这项病毒式营销计划可能存在的问题，以及可能的改进思路，积累这些经验为下一次病毒式营销计划提供参考。

8. 在线游戏手势跟踪

在线游戏公司通常以最细粒度水平记录每位玩家每次点击和移动。这样大量涌入的

“遥测数据”可以实现欺诈检测、对屡战屡败（并因此灰心）的玩家的干预、向将要完成游戏而要离开的玩家提供额外的剧情或游戏目标、新游戏剧情的想法及游戏中新剧情的试验。

9. 社交图谱关系分析

社交图谱是一种表明社交关系的网络图谱，社交网络系统（SNS）通常有 3 种社交关系：一是强关系，即我们关注的人；二是弱关系，即我们被松散连接的人，类似朋友的朋友；三是临时关系，即我们不认识但与之临时产生互动的人。临时关系是人们没有承认的关系，但是会临时性联系的，比如我们在 SNS 中临时评论的回复等。基于大数据分析，能够帮助互联网企业建立起用户的强关系、弱关系甚至临时关系图谱。人立方网站可以帮助我们查询某人的社交图谱，如键入马云，可以获得马云的社交图谱，如图 2-5 所示。

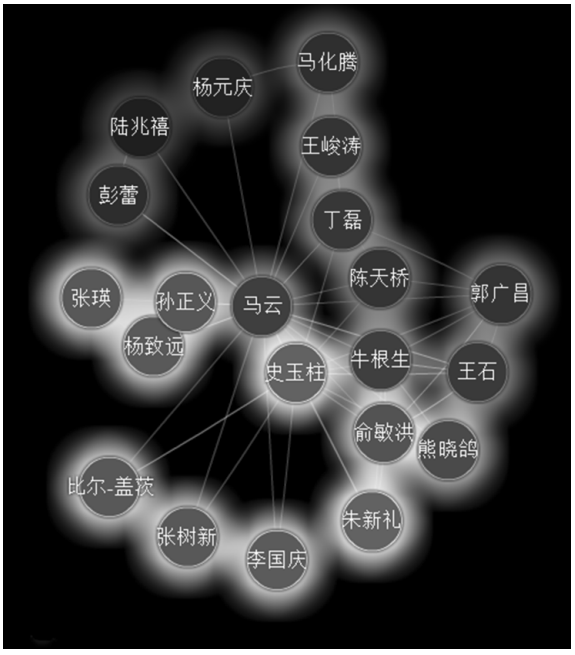


图 2-5　马云的社交图谱

Facebook 从 2013 年 7 月开始向全美用户推出社交图谱搜索（Graph Search），该功能支持用户进行更加高级的搜索，如直接搜索“我朋友喜欢的纽约餐厅”、“我朋友 1996 年之前的照片”等，从而获得详细的搜索结果。

10. 位置和邻近追踪

很多互联网应用在操作应用程序、安全分析、导航和社交媒体中加入了精确 GPS 位置追踪，同时有频繁的更新。精确位置追踪为全球定位系统（GPS）测定点附近其他位置的海量相关数据处理打开了大门，这些其他位置可能带来销售或服务的机会。

2.3.2　零售业

零售行业大数据应用需求目前主要集中在客户行为分析，通过对客户行为的分析，改善货架商品摆放、产品推荐、产品细分和市场营销等。沃尔玛是零售业大数据应用的标杆。

1. 货架商品关联性分析

沃尔玛基于一个庞大的客户交易数据库，对顾客购物行为进行分析，了解顾客购物习惯，发现其中的共性规律。两个著名的应用案例是：“啤酒－尿不湿关联销售”和“手电筒和蛋挞的关联销售”。沃尔玛的大数据分析显示，年轻爸爸一般在买尿不湿的时候，通常要犒劳一下自己，买一打啤酒，因此这两个商品摆放在一起销售的效果很好。另一个是手电筒和蛋挞的例子，沃尔玛的大数据分析显示，在飓风季，手电筒和蛋挞的销量数据都很高。根据这一特点，在飓风这个季节，沃尔玛把手电筒和蛋挞摆在一起可以大幅增加销量。

2. 产品推荐

零售业企业需要根据顾客购买行为的交易数据进行客户群分类，把客户群分为品质性顾客、友善性顾客和理性顾客，并针对不同顾客的诉求进行产品的推荐。如图 2-6 所示为零售业顾客群分析。沃尔玛实验室也开始尝试使用客户的 Facebook 好友喜好和 Twitter 发布的内容来进行数据分析，发现顾客的爱好、生日、纪念日等有价值的信息，进行礼品推荐，实现智能销售。

3. 市场营销

一个典型的零售业大数据分析案例是，美国折扣零售商塔吉特著名的顾客怀孕预测。塔吉特公司分析认为，最会买东西的顾客是妇女，而妇女中的黄金顾客群是孕妇。为了发现顾客中的孕妇，塔吉特通过顾客购买行为的大数据分析找出一些有价值的信息，预测那

些买没有刺激性的化妆品、经常补钙的客户可能是孕妇。根据这一结果，商场把一些孕妇产品广告发送到顾客那里，同时把一些促销品广告也杂七杂八地塞在里面，事实证明，尽管确实有出错的时候，但整体上看，营销效果很好。沃尔玛收购了大数据分析创业公司 Inkiru, 一家专注于大数据的数据分析服务商，帮助公司更加系统地评估和分析客户行为、客户转化率、广告跟踪等，提升市场营销的水平。

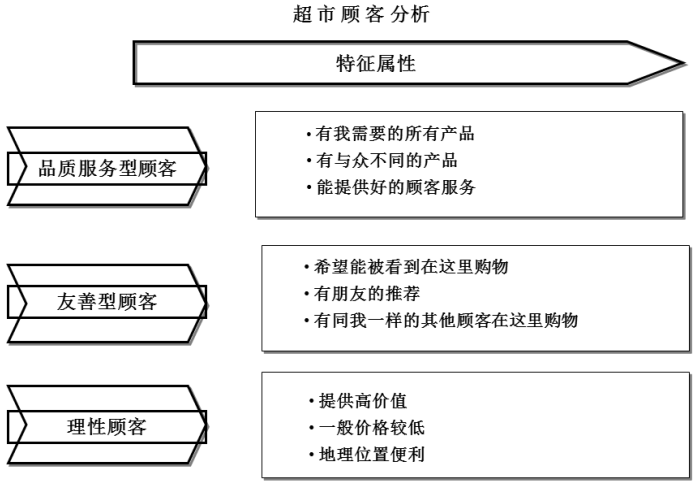


图 2-6　零售业顾客群分类分析

2.3.3　金融业

金融行业应用系统的实时性要求很高，积累了非常多的客户交易数据，金融行业大数据应用的目前主要需求是客户行为分析、金融风险分析等。

1. 基于大数据的客户行为分析

（1）基于客户行为分析的市场营销

招商银行利用客户刷卡、存取款、电子银行转账、微信评论（连接到腾讯公司的数据）等行为数据的研究，每周给顾客发送针对性广告信息，里面有顾客可能感兴趣的产品和优惠信息。花旗银行在亚洲有超过 250 名的数据分析人员，并在新加坡创立了一个“创新实验室”，进行大数据相关的研究和分析。花旗银行所尝试的领域已经开始超越自身的金融

产品和服务的营销。比如新加坡花旗银行会基于消费者的信用卡交易记录，针对性地给他们提供商家和餐馆优惠信息。如果消费者订阅了这项服务，他刷了卡之后，花旗银行系统将会根据此次刷卡的时间、地点和消费者之前的购物、饮食习惯，为其进行推荐。比如此时接近午餐时间，而消费者喜欢意大利菜，花旗银行就会发来周边一家意大利餐厅的优惠信息，更重要的是，这个系统还会根据消费者采纳推荐的比率，来不断学习从而提升推荐的质量。通过这样的方式，花旗银行保持客户的高黏性，并从客户刷卡消费中获益。除花旗外，一些全球信用卡组织也加快了利用大数据的进程。在美国，信用卡企业 Visa 就和休闲品牌商 Gap 合作，来给在 Gap 店附近进行刷卡的消费者提供折扣优惠。美国信用卡企业 MasterCard 分析信用卡用户交易记录，预测商业发展和客户消费趋势，并利用这些结果策划市场营销策略，或者把这些分析结果卖给其他公司受益。

（2）基于客户行为分析的产品创新

我国很多中小企业从银行贷不了款，因为他们没有担保。阿里巴巴公司根据淘宝网上的交易数据情况筛选出财务健康和诚信的中小企业，对这些企业不需要担保就可以贷款。目前阿里巴巴已放贷 300 多亿元，坏账率仅 0.3%。再看一个例子，美国创业公司 ZestCash，主要业务是给那些信用记录不好或者没有信用卡历史的人提供个人贷款服务。ZestCash 的创办人 Douglas Merrill 是谷歌前首席信息官，它和一般银行最大的不同在于其所依赖的大数据处理和分析能力。FICO 信用卡记录得分是美国个人消费信用评估公司开发出的个人信用评级法，大多数美国银行依靠 FICO 分做出贷款与否的决策，这个 FICO 分大概只有 15～20 个变量，诸如信用卡的使用比率、有无未还款的记录等，而 ZestCash 分析的却是数千个信息线索，这形成了它独特的竞争力。例如，如果一个顾客打来电话，说他可能无法完成一次还款，大多数银行会把他视为高风险贷款对象，但是 ZestCash 经过客户相关数据分析发现，这种顾客其实更有可能全额付款，ZestCash 甚至还会考察顾客在提出贷款之前在 ZestCash 网站上停留的时间。

（3）基于客户行为分析的客户满意度分析

花旗银行收集客户对信用卡的质量反馈和功能需求，来进行信用卡服务满意度的评价。质量反馈数据可能是来自电子银行网站或者呼叫中心的关于信用卡安全性、方便性、透支情况等方面的投诉或者反馈，需求可能是关于信息卡在新的功能、安全性保护等方面的新诉求，基于这些数据，他们建立了质量功能屋来进行信用卡满意度分析，并用于服务的优化和改进。如图 2-7 所示为基于大数据分析信用卡业务的客户满意度。

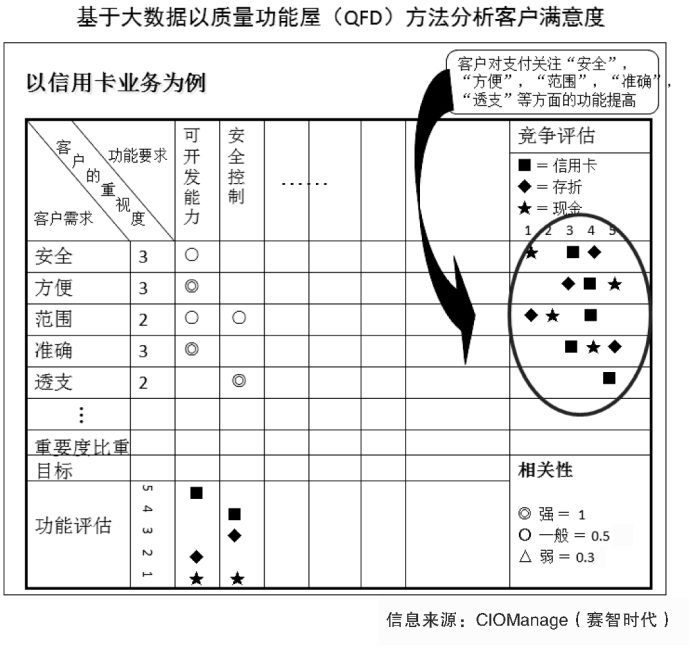


图 2-7　基于大数据分析信用卡业务的客户满意度

（4）投资者情绪分析

华尔街“德温特资本市场”公司对接 Twitter，分析全球 3.4 亿 Twitter 账户流言，判断民众情绪。人们高兴的时候会买股票，而焦虑的时候会抛售股票。依此决定公司股票的买入或卖出，获得较好的收益率。

2. 基于大数据分析的欺诈监测和金融风险管理

（1）金融欺诈行为监测和预防

账户欺诈会对金融秩序造成重大影响。在许多情况下，可以通过账户的行为模式监测到欺诈，在某些情况下，这种行为甚至跨越多个金融系统。例如，“空头支票”需要钱在两个独立账户之间来回快速转账。特定形式的经纪欺诈牵涉两个合谋经纪人以不断抬高的价格售出证券，直到不知情的第三方受骗购买证券，使欺诈的经纪人能够快速退出。金融网站链接分析也能帮助监测电子银行的欺诈。

保险欺诈是全球各地保险公司面临的一个切实挑战。无论是大规模欺诈，例如纵火，

或者涉及较小金额的索赔，例如虚报价格的汽车修理账单，欺诈索赔的支出每年可使企业支付数百万美元的费用，而且成本会以更高保费的形式转嫁给客户。南非最大的短期保险提供商 Santam 通过采用大数据、预测分析和风险划分帮助公司识别出导致欺诈监测的模式，从收到的索赔中获取大数据，根据预测分析及早发现欺诈，根据已经确定的风险因素评估每个索赔，并且将索赔划分为 5 个风险类别，将可能的欺诈索赔和更高风险与低风险案例区分开。

（2）金融风险分析

为评价金融风险很多数据源可以调用，如来自客户经理服务、手机银行、电话银行等方面的数据，也包括来自监管和信用评价部门的数据，在一定的风险分析模型下，帮助金融机构预测金融风险，如图 2-8 所示为金融风险监测和分析。如一笔预期贷款的风险的数据分析，数据源范围就包括偿付历史、信用报告、就业数据和财务资产披露内容等。

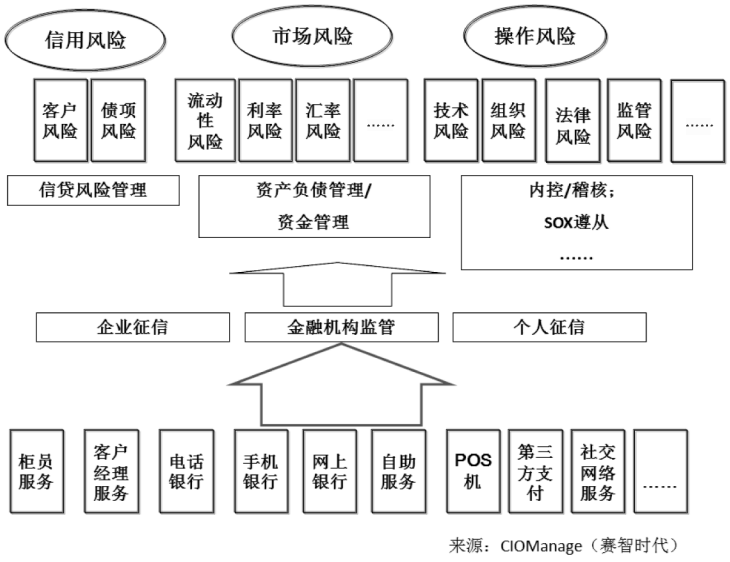


图 2-8　金融风险监测和分析

（3）风险预测

征信机构益百利根据个人信用卡交易记录数据，预测个人的收入情况和支付能力。中英人寿保险公司根据个人信用报告和消费行为分析，来找到可能患有高血压、糖尿病和抑郁症的人，发现客户健康隐患。

2.3.4 政府

政府大数据应用的需求目前有三大方面：一是基于政府数据收集的优势，提供大数据服务，推进政府信息公开；二是基于公众或者企业行为分析，分析和预测经济形势、民主选情、公共服务质量、公共安全监管水平等；三是基于城市物联网数据，对城市基础设施、交通管理、公共安全等方面进行智能化分析和管理。

1. 基于大数据的政府信息公开

DATA.GOV 是美国联邦政府新建设的统一的数据开放门户网站，网站按照原始数据、地理数据和数据应用工具来组织开放的各类数据，截至 2012 年，累积开放 378529 个原始和地理数据集。DATA.GOV 网站上很多数据工具都是公众、公益组织和一些商业机构提供的，这些应用为数据处理、联机分析、基于社交网络的关联分析等方面提供手段。如 DATA.GOV 上提供的白宫访客搜索工具，可以搜寻到访客信息，并将白宫访客与其他微博、社交网站等进行关联，提高访客的透明度。

2. 宏观经济形势的分析和预测

联合国引用美国数据分析软件公司 SAS 的研究数据，以爱尔兰和美国的社交网络活跃度增长作为失业率上升的早期征兆。在社交网络上，网民们更多地谈论“我的车放在车库已经快 2 周了”、“我这周只去了一次超市”、“最近要改坐公共汽车和地铁上班”这些话题时，显示出这些网民可能面临着巨大的失业压力，这些指标是失业预测的领先性指标；当网民开始讨论“我要出租房屋”、“我这个月买了一点点保健品”、“我准备取消到夏威夷的度假”这些话题时，显示出这些网民可能已经失业，面临巨大的生存压力，这些指标是失业后的滞后标志性指标。通过这样的数据分析，帮助政府判断失业形势，提供更多失业救助的政策。如图 2-9 所示为社交媒体情绪数据分析失业率走势。

政府将大数据分析用于经济预测的例子还很多，需求也很大。美国印第安纳大学研究机构利用谷歌公司提供的心情分析工具，从 270 万用户在 2008 年 3 月到 12 月所张贴的 970 万条留言中挖掘出用户的心情，研判 2008 年的金融风暴的社会影响。IBM 日本公司的经济指标预测系统，从互联网新闻中搜索影响制造业的 480 项经济数据，计算出采购经理人指数 PMI 预测值。金融危机前，工业和信息化部委托了一项研究课题，根据阿里巴巴、中国制造网等 B2B 网商的电子商务交易数据，判断电子商务系统能否提前 1～2 月分析预测

宏观经济走势，研究结果表明阿里巴巴出口指数确实对当季海关出口总额的变化有先期预警效果，如图 2-10 所示为阿里巴巴中国出口综合指数与我国出口变动情况比较[3]。

Source: "Can a Country's Online Mood Predict Unemployment Spikes?" *SAS*.

图 2-9　社交媒体情绪数据分析失业率走势

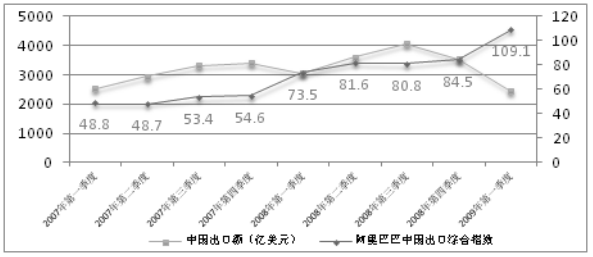


图 2-10　阿里巴巴中国出口综合指数与我国出口变动情况比较

3. 民主选情分析

2012 年，奥巴马利用社交网络和大数据技术预测和分析选民投票动向和竞选走势，为赢得大选做出贡献[4]。

4. 公共安全监测和分析

美国国家安全局和联邦调查局棱镜计划（PRISM）通过进入微软、谷歌、苹果、雅虎

[3]《阿里巴巴中国出口趋势报告 2009 年第一季度》。

[4] http://www.princeton.edu/~rvdb/JAVA/election2012/。

等九大网络巨头的服务器，监控美国公民的电子邮件、聊天记录、视频及照片等资料，名义是保障公共安全、反恐怖。另据报道，美国国家安全局拥有一套基于大数据的新型情报收集系统，名为“无界爆料”系统，以 30 天为周期从全球网络系统中接收 970 亿条信息，通过比对信用卡或通信记录等方式，可以几近真实地还原重点人的实时状况。

5. 城市基础设施实时监测与分析

大数据应用于城市道路桥梁、污染源、大气环境的预测性分析和诊断，即根据道路桥梁传感器获得的大量数据预测性分析常见故障，并根据监测数据比对进行道路桥梁维护。如图 2-11 所示为城市基础设施的实时监测和分析。

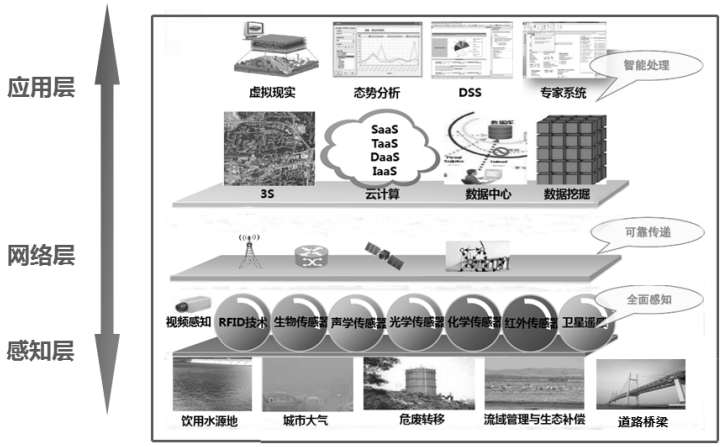


图 2-11　城市基础设施的实时监测和分析

2.3.5　医疗业

医疗行业大数据应用的当前需求主要来自于新兴基因序列计算和分析、基于社交网络的健康趋势分析、医疗电子健康档案分析等领域。

1. 基因组学测序分析

基因组学是大数据在医疗健康行业最经典的应用。基因测序的成本在不断降低，同时产生着海量数据。DNAnexus、Bina Technology、Appistry 和 NextBio 等公司正通过高级算

法和大数据来加速基因序列分析，让发现疾病的过程变得更快、更容易和更便宜。

2. 疫情和健康趋势分析

谷歌公司在官网上有一个利用大数据进行疫情分析的案例[5]。一个地区突然有更多的人通过谷歌来搜索某种疾病，说明这个地区可能处于这种疾病的蔓延期。基于这一假设，谷歌绘制的巴西登革热疫情预测数据与巴西卫生部提供的登革热实际疫情数据基本吻合，充分说明了谷歌基于大数据预测的准确性。如图 2-12 所示为谷歌基于大数据对巴西登革热疫情的预测。

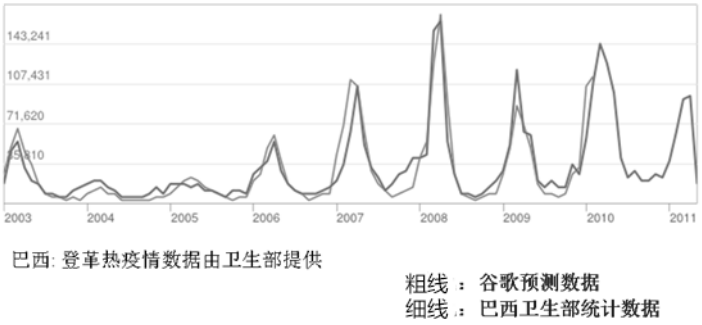


图 2-12　谷歌基于大数据对巴西登革热疫情的预测

在一家名为 Zocdoc 的网站上，求医的病人需要选择医生的专科，Zocdoc 则通过用户选择专科的数据分析，发现不同城市在一个阶段居民对健康领域的关注点，例如“皮肤”，“牙齿”等，以及其他一些信息，从而预测该阶段和该地区的健康趋势。例如，11 月份是流感医生预约最频繁的时段，3 月份是鼻科医生预约高峰期；洛杉矶、拉斯维加斯、凤凰城等城市的居民看急诊的比例（相比例行检查或者预防性治疗）高于其他城市；波特兰市看皮肤病的人最多而费城牙医预约最多。如果，北京市预约挂号平台能够记录和分析这些数据，其实也应该能做这项工作。

3. 医疗电子健康档案分析

一家名为 Apixio 的创业公司正将散布在医院的各个部门、格式各异、标准各异的病历集中到云端，医生可通过语义搜索查找任何病历中的相关信息，从而为医学诊断提供更

[5] http://www.google.org/denguetrends/。

加丰富的数据。CAT 扫描是作为人体“切片”拍摄的图像的堆叠，一家医学大数据分析公司正在对大型 CAT 扫描库进行分析，帮助对医疗问题及其患病率进行自动诊断。

2.3.6 能源业

能源行业大数据应用的需求主要有智能电网应用、跨国石油企业大数据分析、石油勘探资料分析、能源生产安全监测分析等方面。

1. 智能电网应用

在智能电网中，智能电表能做的远不止是生成客户电费账单的每月读数，通过将客户读数频率大幅缩短，例如到每秒每只表一次，可以进行很多有用的大数据分析，包括动态负载平衡、故障响应、分时电价和鼓励客户提高用电效率的长期战略。一家采用智能电表的美国供电公司，每隔几分钟会将区域内用电用户的大宗数据发送到后端集群当中，集群就会对这些数亿条数据进行分析，分析区域用户用电模式和结构，并根据用电模式来调配区域电力供应。在输电和配电端的传感网络能够采集输配电中的各种数据，基于既定模型进行稳态动态暂态分析、仿真分析等为输配电智能调度提供依据。图 2-13 是智能电网的应用架构。

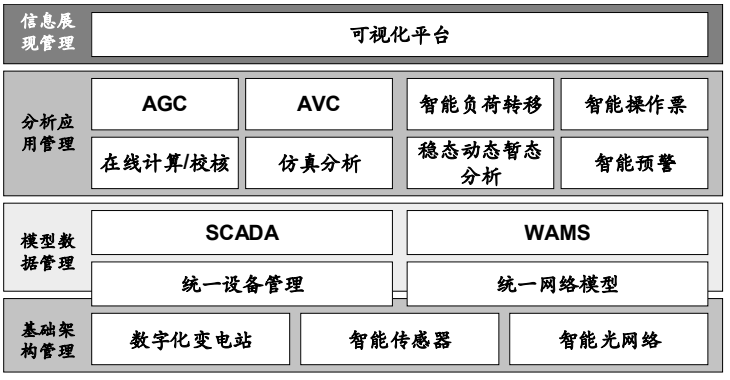


图 2-13 智能电网的应用架构

2. 石油企业大数据分析

大型跨国石油企业业务范围广，涉及勘探、开发、炼化、销售、金融等业务类型，区域跨度大，油田分布在沙漠、戈壁、高原、海洋，生产和销售网络遍及全球，而其 IT 基

础设施逐步采用了全球统一的架构，因此他们已经率先成为了大数据的应用者。例如，雪佛龙公司建立了一个全球的 IT 基础设施结构，称为“全球信息交换网络畅通项目（Global Information Link 1、2、3/Net Ready，缩写为 GIL1、2、3 项目）”，建立全球统一的计算机、网络、服务器标准、存储标准和 IT 服务标准，雪佛龙拥有超过 10 000 台服务器，每天大约新产生 2TB 的数据，每秒新产生 23MB 数据，每天处理 100 万个电子邮件消息。面对海量大数据，雪佛龙公司率先采用 Hadoop 等大数据技术，通过分类和处理海洋地震数据，预测出石油储备状况。

2.3.7　制造业

制造业大数据应用的需求主要是产品需求分析、产品故障诊断与预测、精准营销和工业物联网分析等。

1. 产品需求分析

大数据在客户和制造企业之间流动，挖掘这些数据能够让客户参与到产品的需求分析和产品设计中，为产品创新做出贡献。例如，福特福克斯电动车在驾驶和停车时产生大量数据。在行驶中，司机持续地更新车辆的加速度、刹车、电池充电和位置信息。这对于司机很有用，但数据也传回福特工程师那里，以了解客户的驾驶习惯，包括如何、何时及何处充电。即使车辆处于静止状态，它也会持续将车辆胎压和电池系统的数据传送给最近的智能电话。这种以客户为中心的场景具有多方面的好处，因为大数据实现了宝贵的新型协作方式。司机获得有用的最新信息，而位于底特律的工程师汇总关于驾驶行为的信息，以了解客户，制订产品改进计划，并实施新产品创新。而且，电力公司和其他第三方供应商也可以分析数百万英里的驾驶数据，以决定在何处建立新的充电站，以及如何防止脆弱的电网超负荷运转。

2. 产品故障诊断与预测

无所不在的传感器技术的引入使得产品故障实时诊断和预测成为可能。在波音公司的飞机系统的案例中，发动机、燃油系统、液压和电力系统数以百计的变量组成了在航状态，不到几微秒就被测量和发送一次。这些数据不仅仅是未来某个时间点能够分析的工程遥测数据，而且还促进了实时自适应控制、燃油使用、零件故障预测和飞行员通报，能有效实

现故障诊断和预测。

3. 供应链分析和优化

海尔公司供应链体系很完善，它以市场链为纽带，以订单信息流为中心，带动物流和资金流的运动，整合全球供应链资源和全球用户资源。在海尔供应链的各个环节，客户信息、企业内部信息、供应商信息被汇总到供应链体系中，通过供应链上的大数据采集和分析，海尔公司能够持续进行供应链改进和优化，保证了海尔对客户的敏捷响应，如图 2-14 所示为海尔供应链分析。

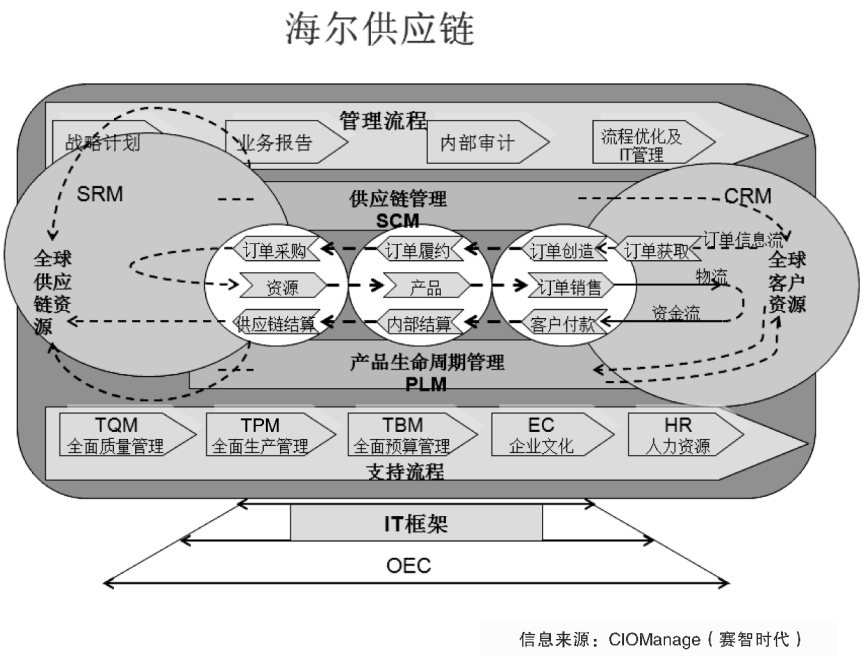


图 2-14　海尔供应链分析

4. 工业物联网分析

现代化工业制造生产线安装有数以千计的小型传感器，来探测温度、压力、热能、振动和噪声。因为每隔几秒就收集一次数据，利用这些数据可以实现很多形式的分析，包括设备诊断、用电量分析、能耗分析、质量事故分析（包括违反生产规定、零部件故障）等。

2.3.8　电信运营业

在网络时代，运营商是数据交换中心，运营商的网络管道、业务平台、支撑系统中每天都在产生大量有价值的数据，基于这些数据的大数据分析为运营商带来巨大的机遇。目前看，电信业大数据应用集中在客户行为分析、网络优化、威胁分析等方面。

1. 客户分析

运营商的大数据应用和互联网企业很相似，客户分析是其他分析的基础。基于统一的客户信息模型，运营商收集来自各种产品和服务的客户行为信息，并进行相应服务改进和网络优化，如图 2-15 所示。如分析在网客户的业务使用情况和价值贡献，分析、跟踪成熟客户的忠诚度及深度需求，包括对新业务的需求，分析、预测潜在客户，分析新客户的构成及关键购买因素（KBF），分析、监控通话量变化规律及关键驱动因素，分析欲换网客户的换网倾向与因素，并建立、维护离网客户数据库，开展有针对性的客户保留和赢回。用户行为分析在流量经营中起重要的作用，用户的行为结合用户视图、产品、服务、计费、财务等信息进行综合分析，得出细粒度、精确的结果，实现用户个性化的策略控制。

某运营商以客户为中心的产品和网络能力分析

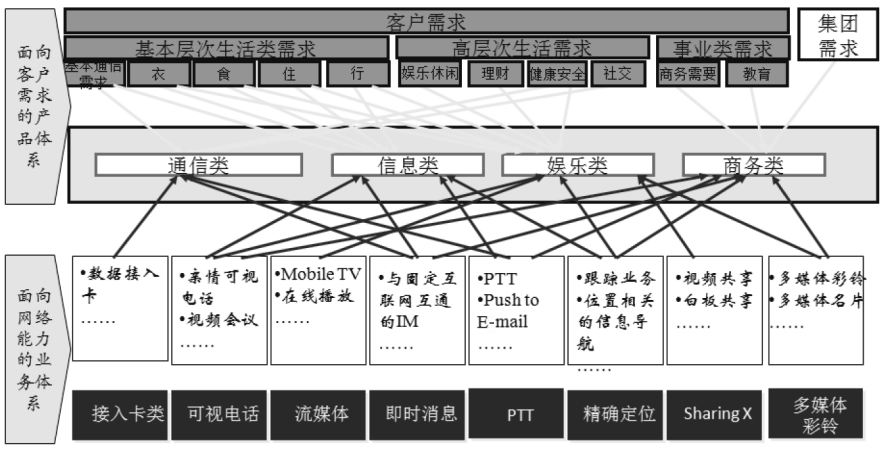


图 2-15　运营商客户分析示意

2. 网络分析与优化

网络管理维护优化是进行网络信令监测，分析网络流量、流向变化、网络运行质量，并根据分析结果调整资源配置；分析网络日志，进行网络优化和故障定义[6]。随着运营商网络数据业务流量快速增长，数据业务在运营商收入占比重不断增加，流量与收入之间的不平衡也越发突出，智能管道、精细化运营成为运营商突破困境的共识。网络管理维护和优化成为精细化运营中的一个重要基础。传统的信令监测尤其是数据信令监测已经面临瓶颈，以某运营商省公司为例，原始数据信令达到 1TB/天，以文件形式保存。而处理之后生成的 xDR（x Detail Record）数据量达到 550GB/天，以数据库形式保存。通常这些数据需要保存数天或数月，传统文件系统以及传统关系数据库处理这么大的数据量显得捉襟见肘。面对信令流量快速增长、扩展困难、成本高的情况，采用大数据技术数据存储量不受限制，可以按需扩展，同时可以有效处理达 PB 级的数据，实时流处理及分析平台保证实时处理海量数据。智能分析技术在大数据的支撑下将在网络管理维护优化中发挥积极作用，网络维护的实时性将得到提升，事前预防成为可能。比如通过历史流量数据以及专家知识库结合，生成预警模型，可以有效识别异常流量，防止网络拥塞或者病毒传播等异常。

3. 安全智能

运营商服务网络的安全监测和预警也是大数据应用的一个重要领域。基于大数据收集来自互联网和移动互联网的攻击数据，提取特征，并进行监测，保障网络的安全。如图 2-16 所示为基于大数据的安全智能。

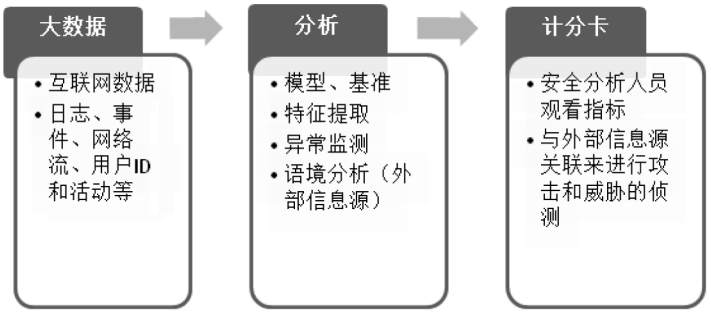


图 2-16　基于大数据的安全智能

[6] 《大数据浪潮中运营商面临的机会》，李秋静　杜学军　叶云，《通信世界》2012 年第 32 期。

2.3.9　交通物流业

交通物流业的大数据应用需求主要是通过大数据的实时分析功能来进行智能交通管理和预测分析，如对违法车辆进行追踪，提高违法车辆追踪的效率；对交通流量进行实时分析和预测，减少道路的拥堵等。

1. 交通流量分析与预测

大数据技术能促进提高交通运营效率、道路网的通行能力、设施效率和调控交通需求分析[7]。例如，根据美国洛杉矶研究所的研究，通过组织优化公交车辆和线路安排，在车辆运营效率增加的情况下，减少 46%的车辆运输就可以提供相同或更好的运输服务。伦敦市利用大数据来减少交通拥堵时间，提高运转效率。当车辆即将进入拥堵地段，传感器可告知驾驶员最佳解决方案，这大大减少了行车的经济成本。大数据的实时性，使处于静态闲置的数据被处理和需要利用时，即可被智能化利用，使交通运行更加合理。大数据技术具有较高的预测能力，可降低误报和漏报的概率，随时针对交通的动态性给予实时监控。因此，在驾驶者无法预知交通的拥堵可能性时，大数据也可帮助用户预先了解。例如，在驾驶者出发前，大数据管理系统会依据前方路线中导致交通拥堵的天气因素，判断避开拥堵的备用路线，并通过智能手机告知驾驶者。北京市就通过交通视频监控数据分析和研判，来确定全市交通状况，并进行智能分析。美国 ComfortDelGro 出租车运营公司后台的数据基础设施能够支持数以十万计的行程、15 000 辆出租车运营数据及数以十亿计的实时 GPS 位置信息。通过分析海量的出租车运营数据，在不同时段和地点推荐能够避免拥堵路段的最佳行车路线预测服务。

2. 交通安全水平分析与预测

大数据技术的实时性和可预测性则有助于提高交通安全系统的数据处理能力。在驾驶员自动检测方面，驾驶员疲劳视频检测、酒精检测器等车载装置将实时检测驾车者是否处于警觉状态，行为、身体与精神状态是否正常。同时，联合路边探测器检查车辆运行轨迹，大数据技术快速整合各个传感器数据，构建安全模型后综合分析车辆行驶安全性，从而可以有效降低交通事故的可能性。在应急救援方面，大数据以其快速的反应时间和综合的决策模型，为应急决策指挥提供辅助，提高应急救援能力，减少人员伤亡和财产损失。

7 《大数据分析：智能交通发展的引擎》，链接：http://www.tranbbs.com/application/platform/application_117190.shtml。

3. 道路环境监测与分析

大数据技术在减轻道路交通堵塞、降低汽车运输对环境的影响等方面有重要的作用。通过建立区域交通排放的监测及预测模型，共享交通运行与环境数据，建立交通运行与环境数据共享试验系统，大数据技术可有效分析交通对环境的影响。同时，分析历史数据，大数据技术能提供降低交通延误和减少排放的交通信号智能化控制的决策依据，建立低排放交通信号控制原型系统与车辆排放环境影响仿真系统。

2.4 企业级大数据应用的共性需求

2.4.1 客户分析

在各个行业中，大数据业务应用需求集中于满足以客户为中心的目标的实现，客户分析是大数据应用的重要领域。企业希望大数据技术有能力更好地了解和预测客户行为，并因此改善客户体验。客户分析的重点是收集和分析交易数据、多渠道交互数据、社交媒体数据、会员卡服务数据及其他与客户相关的数据，以全面提高企业了解客户偏好和需求的能力，真正帮助营销、销售和客户服务部门实现客户关怀的目标。

1. 客户分析的维度

客户分析的基础是客户大数据，CIOManage（赛智时代）咨询公司的研究[8]认为客户大数据分析要包括 3 个重点：一是全面的客户数据分析，二是全生命周期的客户行为数据（包括客户交易数据）分析，三是全面的客户需求数据分析。

- 全面的客户数据——客户是谁？
  - 建立统一的客户信息号和客户信息模型，通过客户信息号，可以获取客户各种相关数据，包括相关业务交易和服务数据。
- 全生命周期的客户行为信息——客户做了什么？
  - 对不同服务、不同生命周期的客户的不同体验进行统一采集、梳理和分析，分析客户行为特点，挖掘客户价值。

[8] 《2013 年中国大数据应用价值研究报告》，CIOManage（赛智时代）公司，2013 年 6 月。

◎　全面的客户需求数据——客户需要什么？

- 收集客户关于产品和服务的新需求，让客户参与产品和服务的创新，促进企业服务的改进和创新。

（1）全面的客户数据分析

客户按照类型可以分为个人客户和企业客户，对应客户的基本信息不同，如个人客户记录姓名、年龄、家庭地址等数据，企业客户记录企业名称、企业注册地、企业法人等数据。从共同的属性来看，有客户基本属性和派生属性，基本属性有客户号、客户类型、客户信用度等，派生属性是由基本属性衍生分析出来的数据，如客户满意度、贡献度、风险度等。客户数据与客户交易数据、客户行为数据、客户需求数据相关联，这种关联关系是通过客户服务的交易、购买的产品、产品厂商、账户等数据来建立的，如图 2-17 所示。

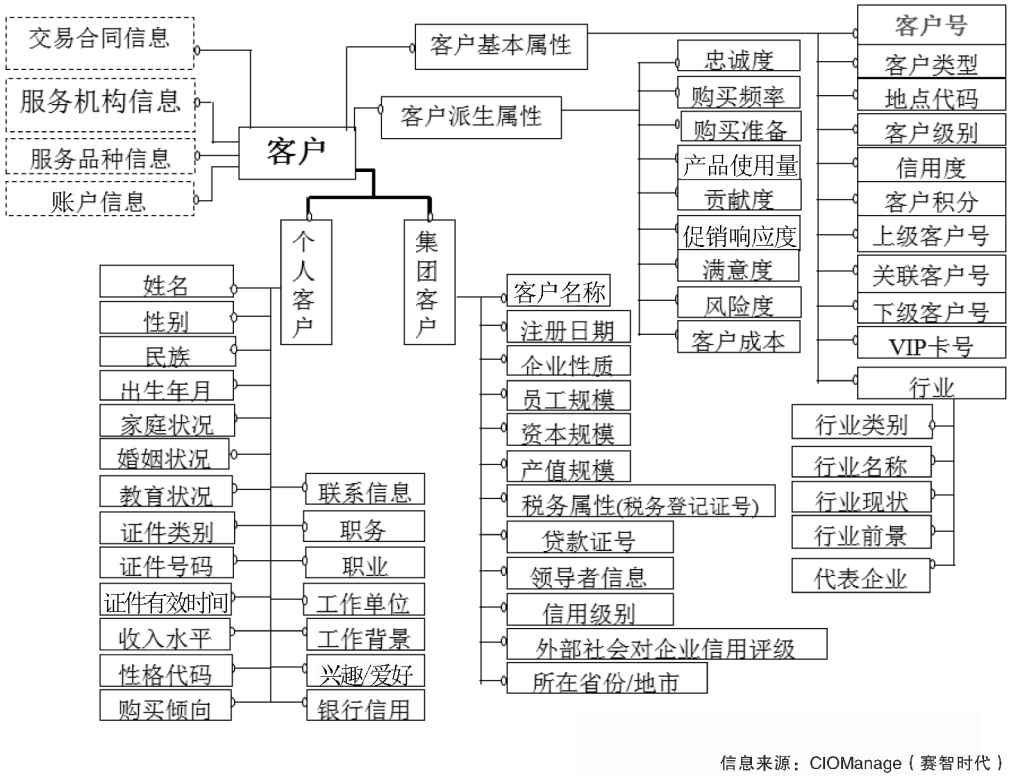


图 2-17　客户数据的模型

（2）全生命周期的客户行为数据分析

客户处于不同生命周期阶段对企业的价值及其需求均有所不同，要求采取不同的管理策略，最大化客户的价值。客户全生命周期分为客户获取、客户提升、客户成熟、客户衰退和客户流失等阶段。在每个阶段，客户需求的特征、客户行为的特征都不同，对这些客户数据的关注度也不同，掌握这些数据，有利于我们在不同阶段掌握不同的客户服务策略。

在客户获取阶段，客户需求特征较为模糊，它的行为模式表现为摸索、了解和尝试阶段。企业需要引导客户，努力把这些客户发展成为潜在客户。这个阶段企业需要关注的客户数据是网上注册的用户资料和访问行为、呼叫中心产品询问信息、网上产品点击流等。通过这些数据的分析，发现客户的潜在需求，并形成与竞争对手的比较优势，通过有效渠道提供合适的价值定位，获取客户。

客户生命周期的行为和需求特征如图 2-18 所示。

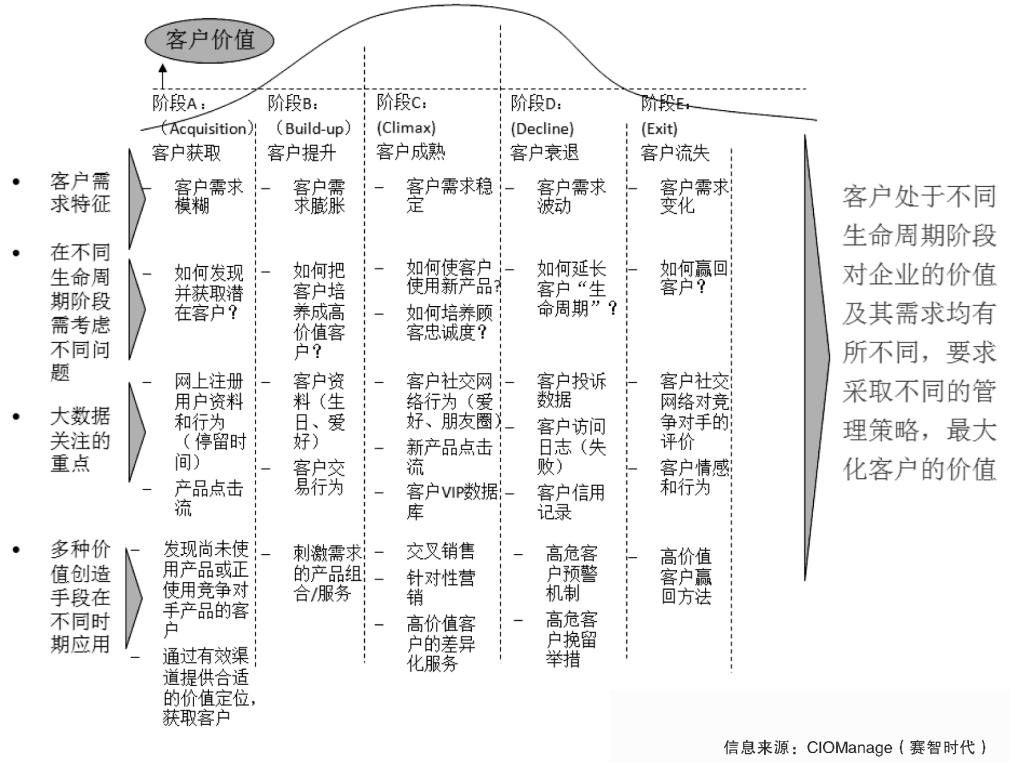


图 2-18　客户生命周期的行为和需求特征

在客户提升阶段，客户已经选购了企业部分产品，经过尝试，客户需求在膨胀。这个阶段客户的行为模式表现为询问产品性价比、查询产品安装指南、发表产品使用评论，寻求产品的增值服务等。这个阶段需要关注的客户行为数据有客户订单、客户资料、产品宣传点击流、产品质量反馈等，特别是要关注客户服务和客户关怀所需要的客户生日、客户喜好、客户职业等信息。企业要采取的对策是把客户培养成为高价值客户，通过不同的产品组合来刺激客户的消费。

在客户成熟阶段，客户成为企业的常客，经常性购买他所喜好的企业产品，客户需求很成熟。这个阶段的行为模式表现为重复购买、与服务部门的信息交流，向朋友推荐自己所使用的产品。在这个阶段，需要关注的客户数据是客户交易数据库、呼叫中心服务记录、客户社交网络行为（爱好、朋友圈）、客户 VIP 数据库等。企业要采取的对策是培养客户忠诚度和新鲜度，激励客户购买新的产品，进行交叉营销，给客户更加差异化的服务。

在客户衰退阶段，客户表现出了对产品的疲倦、情绪波动，甚至不满，开始尝试竞争对手的产品。这个阶段的行为特征是较长时间的沉默，对客户服务进行抱怨，了解竞争对手的产品信息等。这个阶段关注的数据是客户投诉数据、客户访问日志（失败记录）、客户信用记录等。这个阶段企业需要思考如何延长客户生命周期，建立客户流失预警，设法挽留住高价值客户。

在客户流失阶段，客户需求发生了变化，选择了竞争对手的产品。这个阶段的行为特征是放弃企业产品，开始在社交网络上对竞争对手产品给予好评以及对企业产品进行负面评价，这个阶段关注的数据是社交网络数据，特别是客户情绪数据。企业思考的是如何赢回客户，措施可能是客户关怀和让利等。

（3）全面的客户需求数据分析

客户对产品的需求是产品设计的开始，也是产品改进和产品创新的动力源头。收集和分析客户对产品需求的数据，包括外观需求、功能需求、性能需求、结构需求、价格需求等。这些数据可能是模糊的、非结构化的，但对于产品设计和创新而言却是十分宝贵的信息。

2. 客户分析的常见模型

（1）客户分类分析

分析客户属于哪一类的消费客户。例如，可以将客户按照 ABC 的方式进行分类，如图

2-19 所示。这是根据客户贡献度和客户数的维度的统计，主要是挖掘 20%的贡献 80%价值的重要客户。

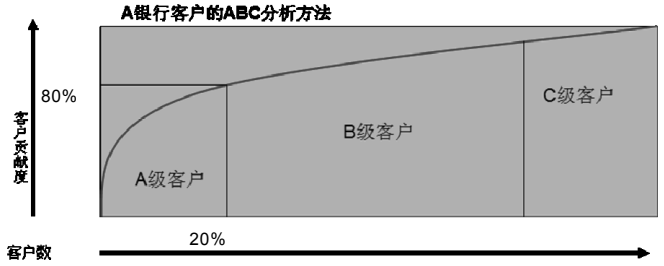


图 2-19　客户 ABC 分类法

（2）社交客户关系管理关键指标分析

Altimeter 咨询公司描述了一套非常有用的社交 CRM 关键绩效指标（KPI）[9]，包括语音共享、观众参与、对话范围、主动倡导、倡导的影响、宣传的影响、解决率、解决时间、满意度评分、主题趋势、情感比和思想影响。这些 KPI 的计算涉及对巨量数据源，尤其是非结构化的社交媒体的深入挖掘。

（3）因果因素发现

销售点数据一直以来都能够向我们表明产品销售何时急剧上升或下降，但是对解释这些背离的因果因素的寻找，却很难令人满意。答案可能会在竞争价格数据，网络、平面和电视媒体在内的竞争促销数据，天气、节日，包括灾害在内的国家大事，以及社交媒体中像病毒一样传播的观点中找到。

（4）客户流失预测

关心客户流失的企业希望了解造成客户流失的预测性因素，包括客户的行为以及很多外部因素，其中包括经济、年龄段和其他人口统计学信息，以及竞争的问题。

2.4.2　绩效分析

企业绩效分析和预测是企业大数据的重要应用之一。企业从内部业务和管理信息系

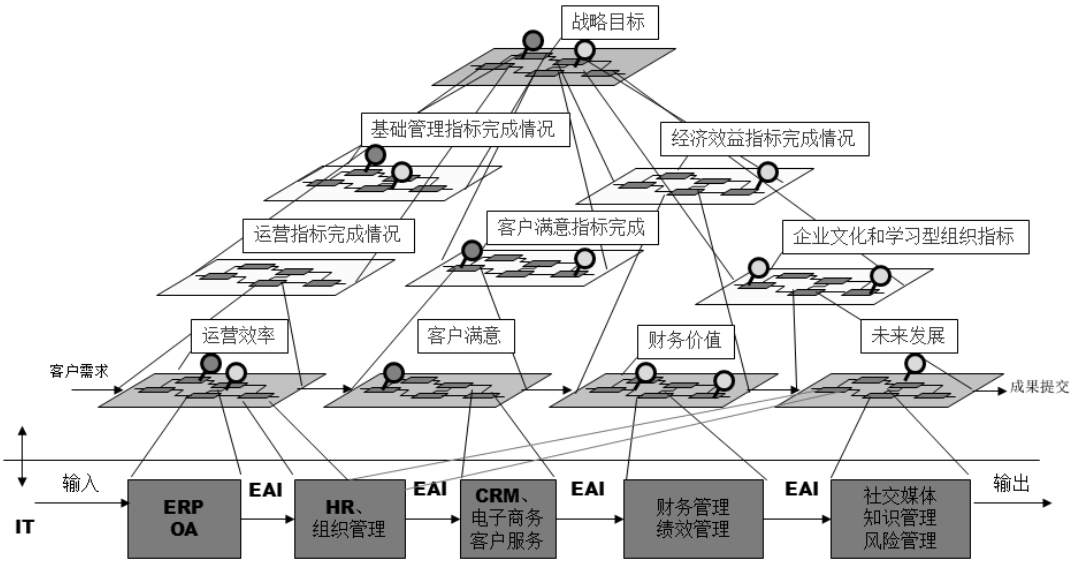
[9] http://www.altimetergroup.com/。

统、外部网站、电子商务和社交媒体等收集了各种类型的数据，通过这些数据分析能够分析和预测企业业务和管理绩效，为企业运行提供全面的洞察力。按照企业平衡计分卡的模型，企业绩效的数据主要包括 4 个方面，分别是企业业务运营数据、财务价值数据、客户数据和面向企业未来发展的数据。

企业业务运营数据主要从企业内部的 ERP 系统、业务系统、生产系统、办公自动化系统等中获取，主要存储在传统的关系型数据库中，当然也能从客户反馈、员工日志、生产线工业数据中分析和获取。财务价值数据主要来自 ERP、财务系统、预算系统等，也能从上市公司年报、政府统计数据、行业年鉴等中获取有用信息。客户数据主要来自内部 CRM 系统、呼叫中心、门户网站、社交媒体，这部分在上一节已经阐述过。

面向企业未来发展的数据来自企业社区、知识管理、人力资源管理、企业即时通信、企业微博等系统，也有来自社会公益组织、政府的数据。

通过企业内外部数据的采集和分析，能够实时反映企业战略目标的执行情况、差距，并对未来战略目标的实现进行提前的预测和分析，如图 2-20 所示。



信息来源：CIOManage（赛智时代）

图 2-20　数据驱动的企业绩效分析

2.4.3 欺诈和风险评估

企业风险管理的大数据应用需求主要是对风险点数据监测、安全隐患提前分析发现、风险提前预警等。

按照企业全面风险管理的模型，要对企业总部、各个业务部门、各个分支机构在内部应用系统、网络、移动终端上的操作内容和行为进行风险监控和数据采集，针对具有专门互联网和移动互联网业务的部门也要对其操作内容和行为进行专门的数据采集。企业也要敏感地获取有关市场风险、信用风险和法律政策风险等数据，为防范这些风险提供预警机制。数据采集主要回答的问题是：关注这些机构包括哪些经营活动？各经营活动中存在哪些风险？如何记录或采集风险数据？风险产生的原因是什么？其中哪些风险是重要的？获得这些数据后，要对风险进行评估和分析，主要关注风险发生的概率大小、风险概率分析情况、风险估算。要按照风险模型，设立风险阈值，一旦超过阈值及时预警和处理。在风险分析的基础上，要进行风险规划，制定减轻风险、转移风险、规避风险、自我消化风险等策略，并选择最佳方案，检验各风险因素对研究指标的影响，最终做出风险处理措施。如图 2-21 所示为数据驱动的企业风险管理。

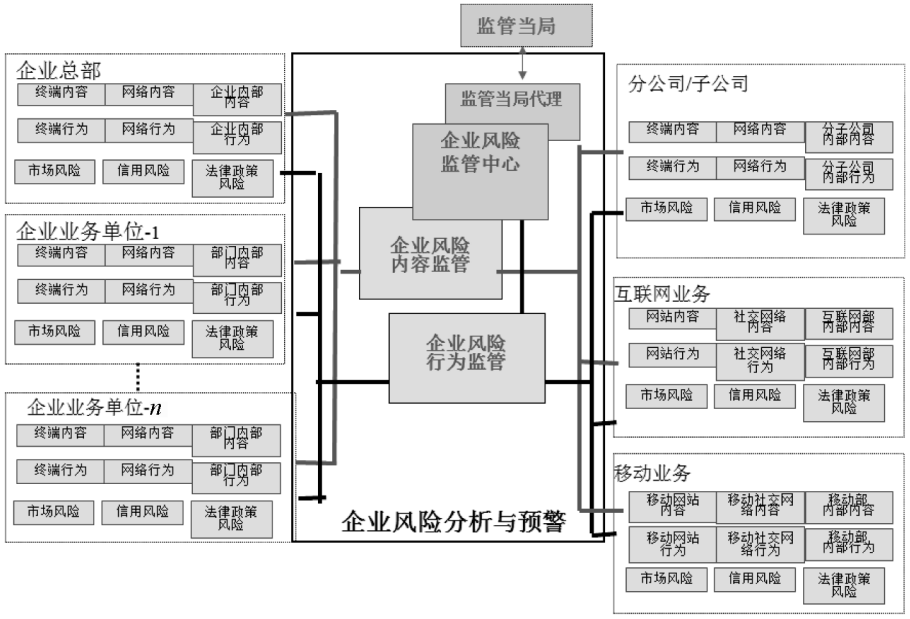


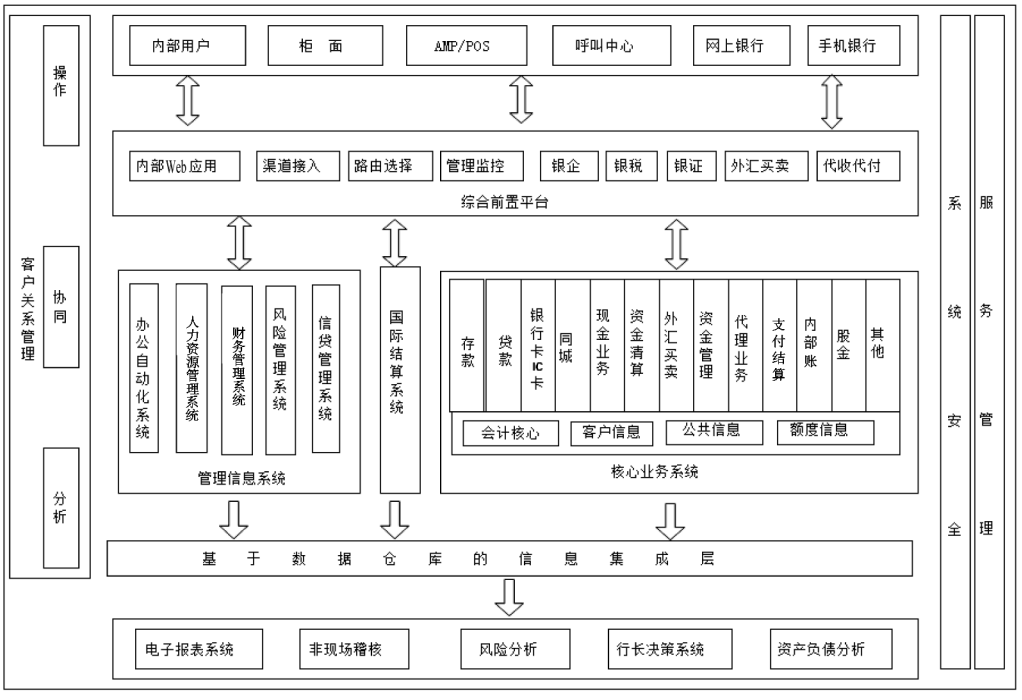
图 2-21 数据驱动的企业风险管理

2.5　以银行客户分析为例，分析一个大数据的应用场景

本书将以银行基于大数据的客户分析的例子为线索，阐述大数据技术如何在这个组织内应用和实施。

我们权且把这家银行称为 A 银行，它是一家股份制商业银行，在全国各地设有分支机构，业务覆盖了存、贷和各类中间业务，客户包括个人客户和企业客户。

A 银行的信息化水平很高，已经在企业内部建立了新一代电子银行核心业务系统、网上银行、信用卡系统、电话银行、移动银行、中间业务系统、管理信息化系统和资产负债管理系统等，形成了较为完善的银行企业信息化架构，如图 2-22 所示。

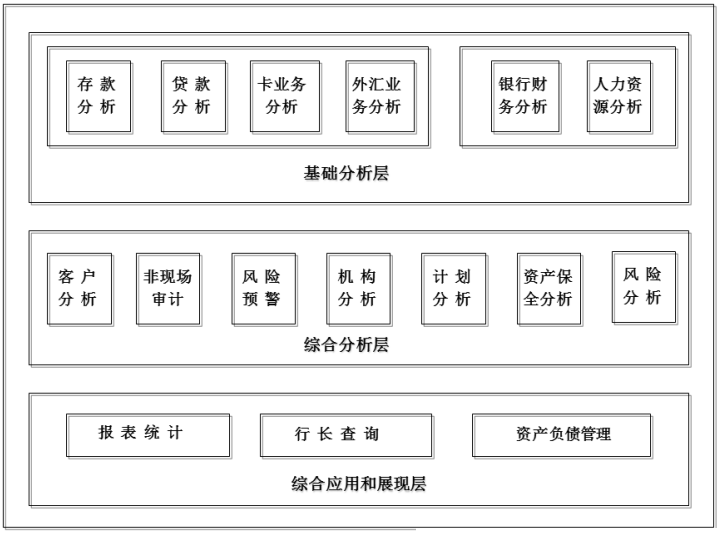


信息来源：CIOManage（赛智时代）

图 2-22　某商业银行信息化架构

A 银行的数据量很大，每天产生的银行业务交易数据存储在 Oracle 数据库中，管理

信息化系统的数据也存储在各自独立的数据库中，数据库品牌主要是Oracle和SQL Server。2012年，A银行开展了主数据管理、数据仓库和商业智能的建设工作，形成了企业级的数据模型、数据仓库和分析架构，如图2-23所示。商业智能分析工具是SAS公司的产品。



信息来源：CIOManage（赛智时代）

图2-23　某银行商业智能系统功能架构

当前，A银行启动了“基于大数据的新一代商业银行客户分析系统项目”，主要业务需求如下。

一、收集电话银行、网络银行、手机银行、信用卡、贷记卡、企业微博等服务渠道的非结构化数据，并基于客户分析模型，进行装载、存储和分析计算。

二、整合数据仓库和商业智能中客户分析系统，利用商业智能分析工具实现分析的可视化，形成报表和分析报告。

三、客户分析的维度主要包括客户资料分析、客户行为分析。客户分析的模型包括满意度模型、客户贡献度模型、客户分类模型（VIP客户、普通客户、潜在客户、流失客户等）、客户信用评估、ABC分析、QFD分析等，支持客户特色服务推荐、流失客户预测等。

第 3 章

大数据应用的总体架构和关键技术

本章阐述大数据应用的总体架构，以及存储、处理和分析的关键技术。

3.1 总体架构

3.1.1 业务目标

对第 2 章分析的大数据应用需求进行总结，在大数据产生、聚集、分析和利用的各个阶段都提出了一些需求。这些需求需要通过大数据技术来实现，这能从架构层面实现业务需求向技术要求的映射，如表 3-1 所示为大数据应用的业务－技术的逻辑映射。

表 3-1　大数据应用的业务—技术的逻辑映射

业务环节	业务需求	技术实现
产生	大数据爆炸 • 数据容量：每 18 个月就翻一番 • 数据类型：多于 80%的数据来自非结构化数据 • 数据速度：数据来源不断变化，数据快速流通	采用一个统一的大数据处理方法，使得企业用户能够快速处理和加载海量数据，能够在统一平台上对不同类型的数据进行处理和存储
聚集	管理大数据的复杂性，需要分类、同步、聚合、集成、共享、转换、剖析、迁移、压缩、备份、保护、恢复、清洗、淘汰各种类型数据	一个数据集成和管理平台，集成各种工具和服务来管理异构存储环境下的各类数据
分析	当前数据仓库和数据挖掘技术，擅长分析结构化的事后数据，在大数据环境下要求能够分析非结构化数据，包括流文件，并能进行实时分析和预测	建立一个实时预测分析解决方案，整合结构化的数据仓库和非结构化的分析工具
利用	满足不同的用户对大数据的实时的多种访问方式	任何时间、任何地点、任何设备上的集中共享和协同
	需要理解大数据怎样影响业务，怎样转化为行动	对大数据影响业务和战略进行建模，并利用技术来实现这些模型

大数据应用的总体架构被业务需求逐步勾勒出来：大数据应用需要采用一个统一的大数据平台，使得用户能够快速处理和加载海量数据，能够在统一平台上对不同类型的数据进行处理和存储，需要采用一个数据集成和管理平台，集成各种工具和服务来管理异构存储环境下的各类数据，并建立一个实时预测分析解决方案，整合结构化的数据仓库和非结构化的分析工具，在平台上用户可以在任何时间、任何地点通过任何设备进行大数据的集中共享和协同访问，大数据应用总体架构能够支撑组织对新的业务战略进行建模，提升组织的洞察力。

3.1.2　架构设计原则

企业级大数据应用架构需要满足业务的需求：一是要求能够满足基于数据容量大、数据类型多、数据流通快的大数据基本处理需求，能够支持大数据的采集、存储、处理和分析；二是要能够满足企业级应用在可用性、可靠性、可扩展性、容错性、安全性和保护隐私等方面的基本准则；三要能够满足用原始技术和格式来实现的数据分析的基本要求，如图 3-1 所示。

图 3-1　大数据架构设计的原则

1. 满足大数据 V3 的要求

◎　大数据容量的加载、处理和分析

要求大数据应用平台经过扩展可以支持 GB、TB、PB、EB 甚至 ZB 的数据集。

◎　各种类型数据的加载、处理和分析

支持各种各样的数据类型，支持处理交易数据、各种非结构化数据、机器数据以及其他新结构数据。支持极端的混合工作负载，包括数以千计地理上分布的在线用户和程序，这些用户和程序执行各种各样的请求，范围从临时性的请求到战略分析的请求，同时以批量或流的方式加载数据。

◎　大数据的处理速度

在很高速度（GB/秒）的加载过程中集成来自多个来源的数据。对高度限定的标准 SQL 查询的亚秒级响应时间。以至少每秒千兆字节的速度高速加载数据，随时进行分析。以满负荷速度就地更新数据。不需要预先将维表与事实表群集即可将十亿行的维表加入万亿行的事实表。在传入的加载数据上实时执行某些“流”分析查询。

2. 满足企业级应用的要求

◎ 高可扩展性

要求平台符合企业未来业务发展要求以及对新业务的响应，能够支持大规模数据计算的节点可扩展，能适应将来数据结构的变化、数据容量增长、用户的增加、查询要求和服务内容的变化，要求大数据架构具备支持调度和执行数百上千节点的复杂工作流。

◎ 高可用性（容错）

要求平台能够具备实时计算环境所具备的高可用性，在单点故障的情况下能够保证应用的可用性，具备处理节点故障时的故障转移和流程继续的能力。

◎ 安全性和保护隐私

系统在数据采集、存储、分析架构上，保证数据、网络、存储和计算的安全性，具备保护个人和企业隐私的措施。

◎ 开放性

要求平台能够支持计算和存储分布到数以千计的、地理位置可能不同的、可能异构的计算节点，能够识别和整合不同技术和不同厂商开发的工具和应用，能够支持移动应用、互联网应用、社交网络、云计算、物联网、虚拟化、网络、存储等多种计算设备、计算协议和计算架构。

◎ 易用性

系统功能操作是否易用，能否满足大多数企业业务、管理和技术人员的操作习惯。平台具有可编程性，能够支持不同编程工具和语言的集成，具备集成编译环境。能否在处理请求内嵌入任意复杂的用户定义函数（UDF），以各种行业标准过程语言执行 UDF， 组合大部分或全部使用案例的大量可复用UDF 库，几分钟内对PB级大小的数据集执行UDF“关系扫描”。

3. 满足对原始格式数据进行分析的要求

系统具备对复杂的原始格式数据进行整合分析的能力，如对文本数据、数学数据、统计数据、金融数据、图像数据、声音数据、地理空间数据、时序数据、机器数据等进行分析的能力。

3.1.3　总体架构参考模型

基于 Apache 基金会开源技术的大数据平台总体架构参考模型如图 3-2 所示，大数据的产生、组织和处理主要是通过分布式文件处理系统来实现的，主流的技术是 Hadoop+MapReduce，其中 Hadoop 的分布式文件处理系统（HDFS）作为大数据存储的框架，分布式计算框架 MapReduce 作为大数据处理的框架。

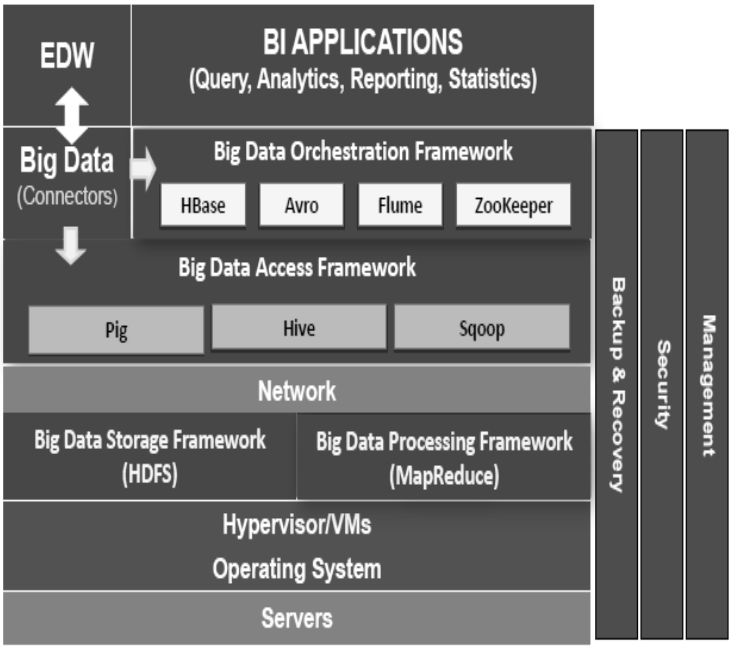


图 3-2　大数据应用平台的总体架构参考模型

1. 大数据存储的框架

HDFS：即 Hadoop 分布式文件系统，分布式文件系统运行于大规模集群之上，集群使用廉价的普通机器构建，整个文件系统采用的是元数据集中管理与数据块分散存储相结合的模式，并通过数据的复制来实现高度容错。分布式文件处理系统架构在通用的服务器、操作系统或虚拟机上。

2. 大数据处理的框架

MapReduce：一个分布式并行计算软件框架，基于 Map（可理解为“任务分解”）和 Reduce（可理解为“综合结果”）的 Java 函数，基于 MapReduce 写出来的应用程序能够运行在由上千个普通机器组成的大型集群上，并以一种可靠容错的方式并行处理 TB 级别以上的数据集。Mapper 和 Reducer 的主代码，可以用很多语言书写。Hadoop 的原生语言是 Java，但是 Hadoop 公开 API 用于以 Ruby 和 Python 等其他语言编写代码。提供了与 C++ 的接口，其名称为 Hadoop Pipes。在最低层次进行 MapReduce 编程显然提供了最大的潜力，但这种编程层次非常像汇编语言的编程，属于低级的编程语言。

3. 大数据访问的框架

在 Hadoop+MapReduce 之上，架构的是网络层。网络层之上，是大数据访问的框架层。大数据访问的框架实现对传统关系型数据库和 Hadoop 的访问，主流技术包括 Pig、Hive、Sqoop 等。

Pig：是基于 Hadoop 的并行计算高级编程语言，它提供一种类 SQL 的数据分析高级文本语言，称为 Pig Latin，该语言的编译器会把类 SQL 的数据分析请求转换为一系列经过优化处理的 MapReduce 运算。Pig 支持的常用数据分析主要有分组、过滤、合并等。Pig 为创建 Apache MapReduce 应用程序提供了一款相对简单的工具，它有效简化了编写、理解和维护程序的工作，还优化了任务自动执行功能，并支持使用自定义功能进行接口扩展[1]。

Hive：由 Facebook 贡献的数据仓库工具，是 MapReduce 实现的用来查询分析结构化数据的中间件。Hive 的类 SQL 查询语言——Hive QL 可以查询和分析储存在 Hadoop 中的大规模数据。

Sqoop：由 Cloudera 开发，是一种开源工具，用于在 Hadoop 与传统的数据库间进行数据的传递，允许将数据从关系源导入 HDFS 以及从 HDFS 导出到关系型数据库。MapReduce 等函数都可以使用由 Sqoop 导入 HDFS 中的数据。

Cascading：Cascading 是另一个工具，是编写复杂 MapReduce 应用程序的抽象层。它的最恰当描述是一个瘦 Java 库，通常被作为查询 API 和进程调度器使用的命令行调用。[2]

[1] http://pig.apache.org/。

[2] http://www.cascading.org/。

4. 大数据调度的框架

在大数据访问的框架层之上是大数据控制调度的框架，实现对大数据的组织和调度，为大数据分析做准备，主流技术包括 HBase、Avro、Flume、ZooKeeper 等。

HBase：是一个基于列存储的开源非关系型数据库（NoSQL 数据库），类似于 BigTable，是 Key-Value 数据库系统。HBase 直接运行在 Hadoop 上。HBase 不是一种 MapReduce 的实现，它与 Pig 或 Hive（MapReduce 的实现）的主要区别是能够提供非常大数据集的实时读取和写入随机存取。

Avro：新的数据序列化格式与传输工具，将逐步取代 Hadoop 原有的 IPC 机制。

ZooKeeper：分布式锁设施，是一个分布式应用程序的集中配置管理器，用于分布式应用的高性能协同服务，由 Facebook 贡献。也可以独立于 Hadoop 使用。

Flume：Cloudera 开发和提供的日志收集系统。Flume 提供一个可靠的分布式流数据收集服务。

Scribe：由 Facebook 开发，作为开源程序发布，用于聚合来自很大数量 Web 服务器的日志数据。

Oozie：是一个基于服务器的工作流引擎，专门调度和运行执行 Hadoop 作业（如 MapReduce、Pig、Hive、Sqoop、HDFS 操作）的工作流。

集成开发环境（IDE）：MapReduce/Hadoop 开发需要摆脱纯手工编码。MapReduce/Hadoop 的集成开发环境需要包括源代码编辑器、编译器、自动化系统构建工具、调试器和版本控制系统。

集成应用程序环境：比集成开发环境还要高的层次，可以被称为集成应用程序环境，在这里通过图形用户界面将复杂的可复用分析路线组装成完整的应用程序。这种类型的环境可能使用开源算法，例如 Mahout 项目提供的算法，将机器学习算法分布在 Hadoop 平台上。

5. 大数据分析和展现框架

大数据分析和展现则通过相关的商业智能分析和展现工具集来实现。

Mahout：Apache Mahout 项目提供分布式机器学习和数据挖掘库。

Hama：基于 BSP 的超大规模科学计算框架。

6. 大数据连接器

大数据分析平台需要建立与传统关系型数据库、数据仓库的连接，大数据连接器的作用是实现大数据平台和传统数据平台的结合。

ETL 平台：ETL 平台在关系型数据库的数据导入和导出方面有很长的历史，为数据进出 HDFS 提供了专门的接口。基于平台的方法与手工编码截然不同，提供了对元数据、数据质量、文档以及系统构建的可视化风格的广泛支持。

7. 大数据管理、安全和备份恢复框架

大数据的管理、安全和备份恢复框架帮助进行大数据治理、安全性和大数据保护。

Ambari：Hadoop 管理工具，可以快捷地监控、部署、管理集群。

Chukwa：用于管理大规模分布式集群的数据收集系统。

嵌入的 Hadoop 管理功能：Hadoop 支持全面的运行时环境，包括编辑日志、安全模式运行、日志审计、文件系统检查、数据节点块验证、数据节点块分布均衡器、性能监控、综合日志文件、管理员度量、MapReduce 用户计数、元数据备份、数据备份、文件系统均衡器、节点启用和弃用。

Java 管理扩展：一种用于监控和管理应用程序的标准 Java API。

GangliaContext：一种用于超大集群的开源分布式监控系统。

3.1.4 总体架构的特点

大数据技术架构具备集成性、架构先进性和实时性等特点。

1. 统一、开放、集成的大数据平台

- 可基于开源软件实现 Hadoop 基础工具的整合。
- 能与关系型数据库、数据仓库通过 JDBC/ODBC 连接器进行连接。
- 能支持地理分布的在线用户和程序，并行执行从查询到战略分析的请求。
- 用户友好的管理员平台，包括 HDFS 浏览器和类 SQL 查询语言等。

- 提供服务、存储、调度和高级安全等企业级应用的功能。

2. 低成本的可扩展性

- 支持大规模可扩展性，到 PB 级源数据。
- 支持极大的混合工作负载，各种数据类型包括任意层次的数据结构、图像、日志等。
- 节点间无共享（sharing-nothing）的集群数据库体系架构。
- 可编程和可扩展的应用服务器。
- 简单的配置、开发和管理。
- 以线性成本扩展并提供一致的性能。
- 标准的普通硬件。

3. 实时地分析执行

- 在声明或发现数据结构之前装载数据。
- 能以数据全载入的速度来准确更新数据。
- 可调度和执行复杂的几百个节点的工作流。
- 在刚装载的数据上，可实时执行流分析查询。
- 能以大于每秒 1GB 的速率来分析数据。

4. 可靠性

- 当处理节点失效时，自动失效恢复和流程连续，不需要操作中断。

3.2　大数据存储和处理技术

3.2.1　Hadoop：分布式存储和计算平台

1. Hadoop 概述

Hadoop 是一个分布式文件系统和并行执行环境，可以让用户便捷地处理海量数据。

Hadoop 这个名字不是一个缩写，它是一个虚构的名字。该项目的创建者，Doug Cutting 解释 Hadoop 的得名："这个名字是我孩子给一个棕黄色的大象玩具起的名字。我的命名标准就是简短，容易发音和拼写，没有太多的意义，并且不会被用于别处。小孩子恰恰是这方面的高手。"Hadoop 由 Apache 软件基金会于 2005 年秋天作为 Lucene 的子项目 Nutch 的一部分正式引入，它受到最先由谷歌实验室开发的谷歌文件系统（GFS）和 MapReduce 的启发。2006 年 3 月，Nutch 分布式文件系统（NDFS）和 MapReduce 分别被纳入 Hadoop 的项目中。

Hadoop 的特点在于能够存储并管理 PB 级数据，能很好地处理非结构化数据，擅长大吞吐量的数据处理，应用模式为"一次写多次读"存取模式。由于采用分布式架构，Hadoop 具有很好的可扩展性（最大节点数不断上升）和容错性。Hadoop 并不擅长存储小文件、有大量的随机读，以及需要对文件进行修改等场合。

Hadoop 的 6 个优势如表 3-2 所示。

表 3-2　Hadoop 的 6 个优势

特　点	描　述
最易于扩充的分布式架构	Hadoop 运行在普通的硬件设备集群上，这些硬件就是基于 X86 架构的普通 PC 服务器或者刀片服务器，硬件被软件松散地耦合在一起。Hadoop 的大数据处理能力是通过大量计算节点的横向扩充来实现的。横向扩充（scale-out）是指计算能力的扩充，是通过增加计算节点的数量来实现的；而纵向扩充（scale-up）是指计算能力的扩充，是通过增加单个计算节点的中央处理器（CPU）和存储等的处理能力来实现的。对 Hadoop 而言，扩充是很容易的工作，即简单增加机架，并告诉系统用新增加的硬件来重新均衡系统。用 Hadoop，近线的扩充是可能的。Hadoop 架构下，增加节点能实现线性扩充，即增加节点可线性增加存储、查询和加载性能。Hadoop 能支持到 4000 个节点+主节点的并行处理能力。假设每个节点有几十 TB 的处理能力，4000 个节点就能形成 PB 级以上的海量数据处理的能力
善于处理非结构化数据	关系型数据库管理系统（RDBMS）或者并行处理系统（MPP）用提取－转换－装载（ETL）过程来实现数据库到数据仓库的转换。这对于预先定义好格式的数据被转换到可预知格式的目标数据是很有效的。但是，ETL 对半结构化和非结构化以及复杂数据并不有效。在这种情况下，Hadoop 基于一个低成本、灵活、高可扩展的分布式文件系统，它能使非结构数据处理从传统数据库"笨拙"的 ETL 工作中解放出来

续表

特　　点	描　　述
自动化的并行处理机制	Hadoop 内部处理自动化并行，无须人工分区或优化。数据分布在所有的并行节点上，每个节点只处理其中一部分数据。每个节点上的数据加载与访问方式与关系型数据库相同。所有的节点同时进行并行处理，节点之间完全无共享，无输入/输出（I/O）冲突，是最优化的 I/O 处理。Hadoop 能够在节点之间动态地移动数据，并保证各个节点的动态平衡，因此处理速度非常快。Hadoop 做过任务并发性能优化，在 Hadoop 运行分析任务时，比同样运行在 RDBMS 和 MPP 上的任务更快
可靠性高、容错强	数据复制被建立在 Hadoop 中。Hadoop 能够自动保存数据的多个副本，并且能够自动将失败的任务重新分配。数据丢失的概率很小，同时保证了数据的存储成本很低
计算靠近存储	在 Hadoop 中，计算与存储是一体的，如图 3-3 所示，计算向数据靠拢，实现了一种高效专用的存储模式，能实现任务之间无共享、无依赖，具有高的系统横向延展性。Hadoop 要分析的数据通常都是 TB 级以上的，网络 I/O 开销不可忽视，但分析程序通常不会很大，所以系统传递的是计算方法（程序），而不是数据文件，因此每次计算在物理上都是在相近的节点上进行的（同一台机器或同局域网），大大降低了 I/O 消耗，而且计算程序如果要经常使用，也可以做缓存
低成本计算和存储	在 Hadoop 中，硬件和存储都是价格很便宜的普通设备，当硬件和存储需要增加时，Hadoop 集群能很方便地增加它，成本也很低廉。所以，在 Hadoop 中，数据保留的时间可以很长。不必采用数据取样进行决策，也就能以更细粒度的全量数据做分析，这可以增加数据分析的准确性。直接从 Hadoop 框架中存储并进行分析，比从后台的近线存储中调出数据更有效率、更节省成本

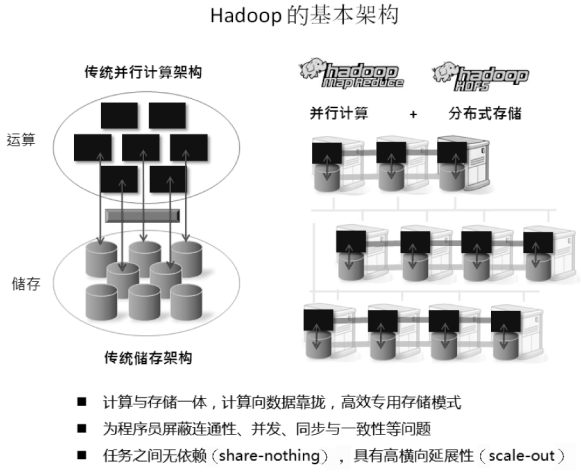


图 3-3　Hadoop 计算与存储架构

2. Hadoop 组成

Hadoop 是 Apache 软件基金会下面的开源项目，主要组成部分有 3 个。

- Hadoop 的分布式文件系统：HDFS。
- Hadoop 并行计算框架：MapReduce。
- 除 HDFS、MapReduce 外的其他项目公共内容：Hadoop Common，支持 Hadoop 的实用程序，包括 FileSystem（面向通用文件系统的抽象基类）、远程程序调用（RPC）和序列化库。

根据 Apache 官方网站提供的 Hadoop 包[3]，其具体组成如表 3-3 所示。

表 3-3　Hadoop 包的组成

包	功能描述
Tool	提供一些命令行工具，如 DistCp、Archive
mapreduce	Hadoop 的 MapReduce 实现
filecache	提供 HDFS 文件的本地缓存，用于加快 MapReduce 的数据访问速度
fs	文件系统的抽象，可以理解为支持多种文件系统实现的统一文件访问接口
Hdfs	HDFS，Hadoop 的分布式文件系统实现
Ipc	一个简单的 IPC 的实现，依赖于 IO 提供的编解码功能
Io	表示层，将各种数据编码/解码，方便于在网络上传输
Net	封装部分网络功能，如 DNS、socket
security	用户和用户组信息
conf	系统的配置参数
metrics	系统统计数据的收集，属于管理员范畴
util	工具类
record	根据 DDL（数据描述语言）自动生成他们的编解码函数，目前可以提供 C++和 Java
http	基于 Jetty 的 HTTP Servlet，用户通过浏览器可以观察文件系统的一些状态信息和日志
Log	提供 HTTP 访问日志的 HTTP Servlet

[3] http://hadoop.apache.org/。

3. Hadoop 与传统数据库（RDBMS）、并行处理系统（MPP）的比较

关系型数据库的管理模型追求的是高度的一致性和正确性。面向超大数据的分析需求，当需要进行扩展时发现，现有架构只能做纵向扩充（scale-up），即通过增加或者更换 CPU、内存、硬盘以扩展单个节点的能力，这种扩展终将遇到瓶颈。关系型数据库不能支持横向扩充（scale-out）。横向扩充是通过增加计算节点连接成集群，并且改写软件，使之在集群上并行执行，这是更为经济的解决办法。而且，关系型数据库在处理大吞吐量的并发操作时，处理时间过长。典型的 RDBMS 有 Oralce、DB2、SQL Server、MySQL 等。

数据仓库中的大规模并行处理系统（Massively Parallel Processing，MPP），基本定义是将任务并行地分散到多个服务器和节点上，在每个节点上计算完成后，将各自部分的结果汇总在一起得到最终的结果。MPP 追求高度的一致性和容错性，即通过分布式事务、分布式锁等机制来实现并发处理，但却无法获得良好的扩展性和系统可用性，而系统的扩展性是大数据分析的重要前提。典型的 MPP 系统有 Oracle 的 Exadata、Teradata Aster Data、IBM 的 Netezza、HP 的 Vertica、EMC 的 Greenplum 等。

表 3-4 是 Hadoop 与 RDBMS 和 MPP 的比较。

表 3-4　Hadoop 与 RDBMS、MPP 的比较

	Hadoop	RDBMS	MPP
数据量	TB->PB	GB->TB	GB->TB
数据类型	所有	结构化数据	结构化数据
事务处理	大量并行处理和 网格处理	串行处理	依旧是串行处理， 但一些处理并行
存取方式	离线批处理	在线交互式与批处理	在线交互式与批处理
数据更新	一次写，多次读	多次读写	多次读写
扩充方式	大量的横向扩充 （Scale-out）	纵向扩充 （Scale-up）	有限的横向扩充 （Scale-out）
数据结构	Key-Value 无 Schema	关系表 固定 Schema	关系表 固定 Schema
编程	函数编程 离线批处理	声明查询 在线事务	声明查询 在线事务
存储	非关系型/NoSQL 数据库	关系型/SQL 数据库	专用的数据仓库和 数据集市

续表

	Hadoop	RDBMS	MPP
资料一致性	低	高（ACID）	高（ACID）*
扩充性	线性	非线性	非线性
分析	不基于模型	基于模型	基于模型
硬件	分布式网格或者集群下的普通硬件	单处理器到多核处理	数据仓库一体机
体系结构	节点间无共享	共享硬盘和内存	无共享

* ACID 是指数据库事务正确执行的 4 个基本要素的缩写，包含原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）。一个支持事务（Transaction）的数据库系统，必须要具有这 4 种特性，否则在事务过程当中无法保证数据的正确性，交易过程极可能达不到交易方的要求。

举一个例子来进行比较。为了说明这些数据管理系统的不同，考虑一个选票计算的应用场景。问题是有 3 个候选人，有 66 亿选票需要被尽可能快地计算，结果必须完全准确。为了保证精确性，每个选票都要被检查。选票计算任务可以用 RDBMS、MPP 或者 Hadoop 来计算。

方法 1：RDBMS

在一个传统的关系型数据库管理系统中（RDBMS），所有选票被保存在一个存储位置，一台机器被指定来进行选票计算，选票计算是顺序进行的。当选民的意图被清楚地识别后，选票被记录，系统移到下一个选票。一旦所有 60 亿选票被计算，结果被送到选举委员会。如果在下次选举中选票增加，RDBMS 能够通过增加存储的数量来进行纵向扩充。

方法 2：MPP

在一个 MPP 系统中，所有选票被保存在一个存储位置，3 个（任意数）计算机节点被指定来处理选票计算。每个节点被发送一个选票批次的虚拟副本，这个批次将被中心计算机计算。每个选票的批次被顺序计算。当选民的意图被识别后，选票被记录，系统移到下一个选票。一旦这些选票被计算完，这些批次再返回到中心计算机，组装结果，并交付到选举委员会。对更大的选举量，MPP 具备有限的能力来横向扩充，即能配备更多的计算节点。

方法 3：Hadoop 框架

用 Hadoop 框架，选票堆被分为 9 个任意的块，每个存储在小的存储位置，而不是一

个大的集中位置。让 9 个官方的投票计算器（计算节点）指定处理选票计算。这些节点与拥有这些数据块的存储放在一起。当选民意图被识别，节点把选票映射到一个键值对，例如候选人 A,1；候选人 B,1；或者候选人 C,1。为了下一个阶段的计算，这个信息被与选票一起保存。一旦每个选票被计算，任务管理器决定键值对怎样以及往哪里传送。对 3 个候选人来说，3 个节点被指定：每个键值对应一个。9 个节点的每一个按照 3 个不同的键来归类所有的结果，并把它们送到相应的节点来进行收集。最终，每个键的和被送到另一个机器来组装结果，并把结果交付到选举委员会。如果下次选举选票数更大，Hadoop 系统能线性扩展，差不多没有限制地按照增加需要的节点以更快地处理数据。毫无疑问，Hadoop 的方法是最低成本、高效和可扩充的解决方案。这个例子很好地解释了大量并行可以通过用 Hadoop 系统处理数据来实现。Hadoop 的节点数可以扩展到成千上万个，来处理更大、更复杂的数据。

3.2.2　Hadoop 之 HDFS：分布式文件系统

1. HDFS 的前身——谷歌文件系统（GFS）

Hadoop 分布式文件系统（HDFS）运行于普通机器构建的大规模集群之上。HDFS 是受谷歌文件系统（GFS）启发构建的。谷歌的数据中心使用廉价的 Linux PC 服务器组成集群，在上面运行各种应用。分布式文件系统 GFS 是谷歌数据中心的存储系统，它隐藏了下层的负载均衡、冗余复制等实现细节，对上层应用程序提供一个统一的文件系统应用程序接口（API）。谷歌根据自身的需求对它进行了特别优化，包括：对超大文件的访问，读操作比例远超过写操作，容错处理以防范 PC 服务器极易发生故障所造成的节点失效等。GFS 把文件分成 64MB 大小的块，块分布存储在集群的机器节点上，用 Linux 的文件系统存放。同时每个块至少有 3 份以上的冗余。中心是一个主节点（Master），根据文件索引，来找寻文件块，GFS 技术细节详见谷歌工程师发布的 GFS 论文[4]。如图 3-4 所示为谷歌的 Linux PC 服务器集群。

[4] http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/zh-CN//archive/gfs-sosp2003.pdf。

图 3-4　谷歌的 Linux PC 服务器集群

2. HDFS 的架构

HDFS 的架构[5]如图 3-5 所示。对外部客户机而言，HDFS 就是一个文件系统，它可以创建、删除、移动或重命名文件等。但是，HDFS 的内部架构是基于一组特定的节点构建的，这些节点包括如下，HDFS NameNode 和 DataNode 的比较如表 3-5 所示。

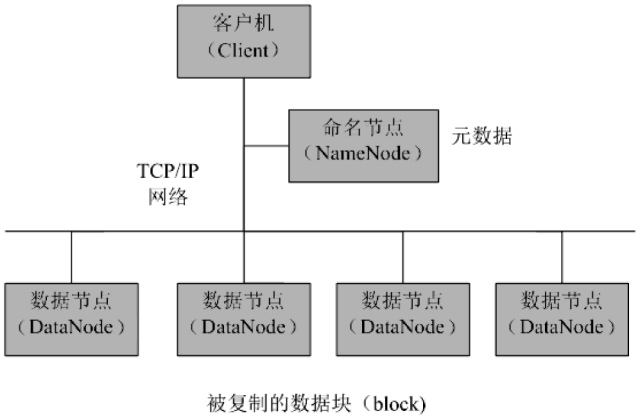


图 3-5　HDFS 的架构图

NameNode（仅一个）：命名节点（也是主节点），它在 HDFS 内部提供元数据服务，用于存储文件的元数据，如文件名、文件目录结构、文件属性（生成时间、副本数、文件权限等），以及每个文件的块列表、块所在的 DataNode 等。NameNode 可以控制所有文件操作。由于仅存在一个 NameNode，因此这是 HDFS 的一个缺点，它可能存在单点故障。这一缺点在新版本的 Hadoop 中有所改善。

[5] http://hadoop.apache.org/core/docs/current/hdfs_design.html。

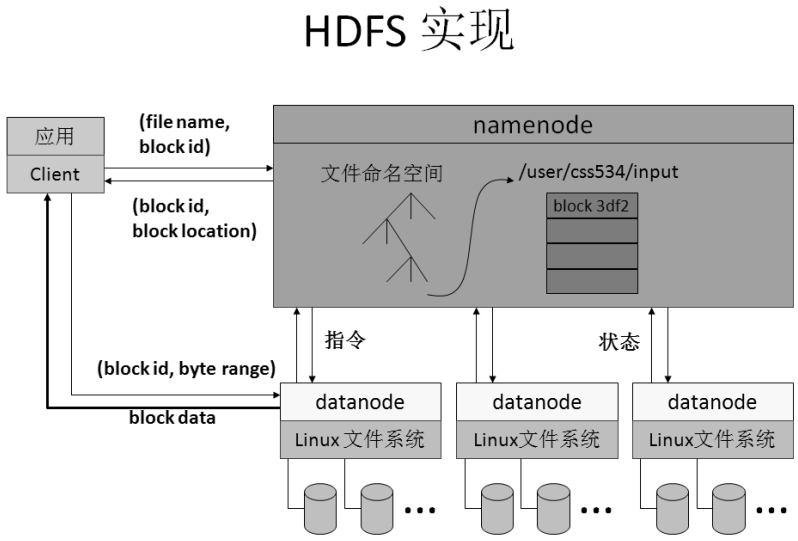
DataNode：数据节点，它为 HDFS 提供存储块。存储在 HDFS 中的文件被分成块，然后将这些块复制到多个计算机中（DataNode）。DataNode 在本地文件系统中存储块数据以及块数据的校验和。这与传统的 RAID 架构大不相同。块的大小（通常为 64MB）和复制的块数量在创建文件时由客户机决定。

表 3-5　HDFS NameNode 和 DataNode 的比较

NameNode	DataNode
• 存储元数据	• 存储文件内容
• 元数据保存在内存中	• 文件内容保存在磁盘
• 保存文件、块、DataNode 之间的映射关系	• 维护了块 ID 到 DataNode 本地文件的映射关系

Hadoop 集群包含一个 NameNode 和大量 DataNode。HDFS 内部的所有通信都基于标准的 TCP/IP 协议。

HDFS 部署在价格低廉的一般计算机硬件的集群上。Hadoop 框架可在单一的 Linux 平台上使用（开发和调试时），但是使用存放在机架上的商业服务器集群才能发挥它的力量。这些机架组成一个 Hadoop 集群。它通过集群拓扑知识决定如何在整个集群中分配作业和文件，如图 3-6 所示。



来源：http://*courses.washington.edu/css534/survey/**Hadoop**File.ppt*

图 3-6　HDFS 的实现

3. HDFS 的文件读写原理

图 3-7 表示了 HDFS 的工作原理。

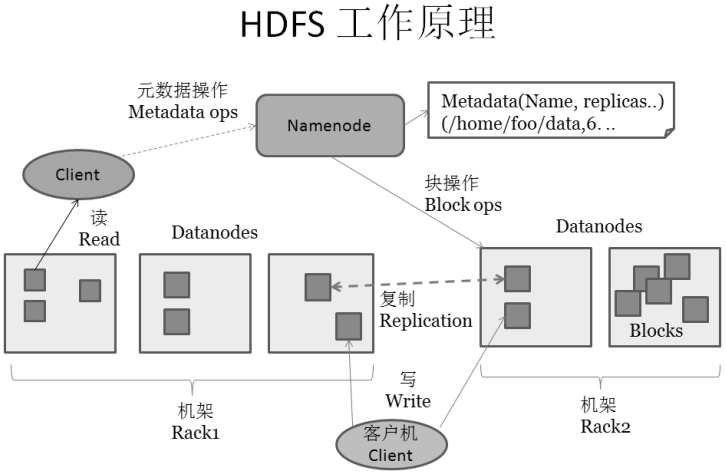


图 3-7　HDFS 的工作机理

（1）NameNode 的运行

NameNode 是在 HDFS 系统中一台单独的机器上运行的软件。它负责管理文件系统名称空间和控制外部客户机的访问。NameNode 决定是否将文件映射到 DataNode 上的各个复制块上。一个文件写到 HDFS 系统中，通常有 3 个复制的块，除自身的块外，还一个复制块存储在同一机架的不同节点上，第 3 个复制块存储在不同机架的某个数据节点上。

实际的输入/输出（I/O）事务并没有经过 NameNode，只有表示 DataNode 和块的文件映射的元数据经过 NameNode。当外部客户机发送请求要求创建文件时，NameNode 会以块标识和该块的第一个副本的 DataNode 的 IP 地址作为响应。NameNode 还会通知其他将要接收该块的副本（复制块）的 DataNode。

NameNode 在一个称为 FsImage 的文件中存储所有关于文件系统名称空间的信息。这个文件和一个包含所有事务的记录文件（例如 EditLog）将存储在 NameNode 的本地文件系统上。FsImage 和 EditLog 文件也需要复制副本，以防文件损坏或 NameNode 系统丢失。

NameNode 本身不可避免地具有单点失效（Single Point Of Failure，SPOF）的风险，

主备模式并不能解决这个问题，目前通过 Hadoop Non-stop NameNode 能实现 100% 正常运行的可用时间[6]。

（2）DataNode 的运行

DataNode 也是在 HDFS 系统中的一台单独机器上运行的软件。DataNode 通常以机架的形式来组织，机架通过一个交换机将所有系统连接起来。Hadoop 的一个假设是：机架内部节点之间的传输速度要快于机架间节点的传输速度。

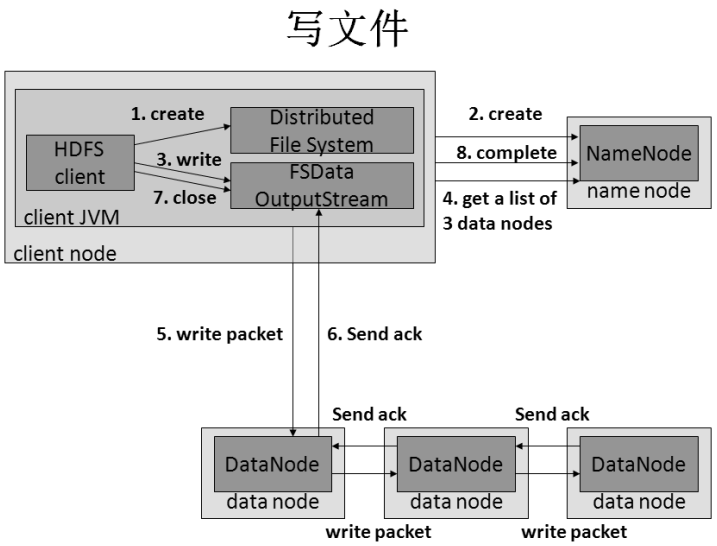
DataNode 响应来自 HDFS 客户机的读写请求。它们还响应来自 NameNode 的创建、删除和复制块的命令。NameNode 依赖来自每个 DataNode 的定期心跳消息（Heartbeat，注：心跳机制在一定时间内即使没有其他消息发送，也要给 NameNode 发送证明存活的心跳消息）。每条消息都包含一个块报告，NameNode 可以根据这个报告验证块映射和其他文件系统元数据。如果 DataNode 不能发送心跳消息，NameNode 将采取修复措施，重新复制该节点上丢失的块。

（3）HDFS 的文件读写操作

HDFS 的主要作用是支持以流的形式访问写入的大型文件。如果客户机想将文件写到 HDFS 上，首先需要将该文件缓存到本地的临时存储。如果缓存的数据大于所需的 HDFS 块大小，创建文件的请求将发送给 NameNode。NameNode 将以 DataNode 标识和目标块响应客户机。同时也通知将要保存文件块副本的 DataNode。当客户机开始将临时文件发送给第一个 DataNode 时，将立即通过管道通信（Pipeline）方式将块内容转发给副本 DataNode。客户机也负责创建保存在相同 HDFS 名称空间中的校验和文件。在最后的文件块发送之后，NameNode 将文件创建提交到它的持久化元数据存储（EditLog 和 FsImage 文件）中。HDFS 可以创建、删除、移动或重命名文件，当文件创建、写入和关闭之后不能修改文件内容。基本写文件操作[7]如图 3-8 所示，HDFS 支持一次写，多次读。基本读文件操作如图 3-9 所示。

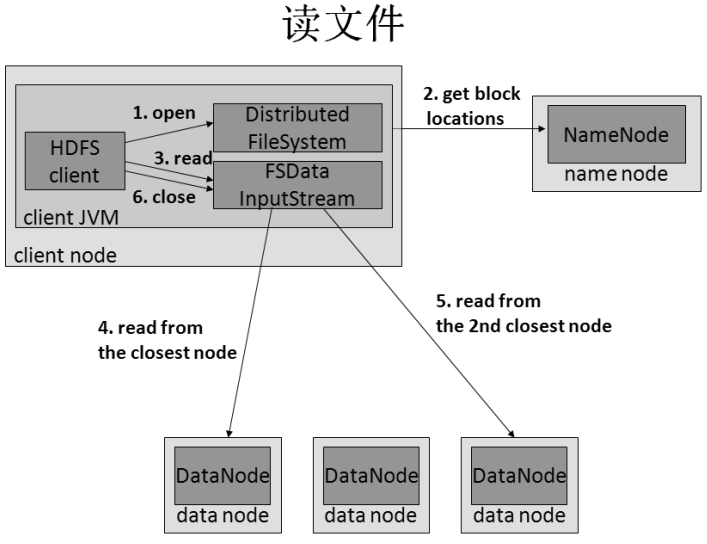
[6] http://www.wandisco.com/。

[7] http://courses.washington.edu/css534/survey/HadoopFile.ppt。



来源：http://*courses.washington.edu/css534/survey/**Hadoop**File.**ppt***

图 3-8　HDFS 写文件操作



来源：http://*courses.washington.edu/css534/survey/**Hadoop**File.**ppt***

图 3-9　HDFS 读文件操作

（4）HDFS 的用户接口

HDFS 的用户接口有 3 种方式。

一是 Java API，通过 MapReduce 访问 HDFS。

二是 Hadoop 的命令行，例如：

```
hadoop dfs -mkdir /foodir
hadoop dfs -cat /foodir/myfile.txt
hadoop dfs -rm /foodir myfile.txt
hadoop dfsadmin -report
hadoop dfsadmin -decommission datanodename
```

三是通过 Web 接口，例如：

```
http://host:port/dfshealth.jsp。
```

下面是一个写文件的例子：

命令行（在 hdfs://master:9000/user/coole 下新建 input 目录）：

```
hadoop fs -mkdir input hdfs://master:9000/user/coole
```

程序接口：

```
//在 hdfs://master:9000/user/coole 目录下创建文件并写入内容
public class DFSOperator {
   public static void main(String[] args) {
      Configuration conf = new Configuration();
      try {
         FileSystem fs = FileSystem.get(conf);
         Path t = new Path("hdfs://master:9000/user/coole/dfs_operator.txt");
             FSDataOutputStream os = fs.create(t,true);
             int i = 0;
             for (i = 0 ;i<5; i++)
             os.writeChars("test");
             os.close();
             } catch (IOException e) {
             e.printStackTrace();
             }
}
}
```

3.2.3 Hadoop 之 MapReduce：分布式计算框架

1. MapReduce 框架概述

Hadoop MapReduce 是一个使用简单的分布式计算软件框架，也是受谷歌 MapReduce 的启发。基于 MapReduce 写出来的应用程序能够运行在由普通机器组成的大型集群上，并以一种可靠容错的方式并行处理 TB 级别以上的数据集。这允许没有任何并行和分布式系统经验的编程者轻松利用一个大型分布式系统中的资源。MapReduce 框架实现的是跨节点的通信，擅长横向扩充、负载均衡、失效恢复、一致性等功能，非常适合有很多批处理的大规模分布式应用，如日志处理、Web 索引建立等。

MapReduce 框架基本原理如图 3-10 所示，大多数分布式运算可以抽象为 Map/Reduce 操作。MapReduce 框架运行在<key,value> 键值对上，Map 是把原始输入 Input 分解成一组中间的<key, value>键值对，Reduce 把结果合成最终输出 Output。

这样一项工作在 MapReduce 被称为作业（job）。一个作业由若干任务（task）组成，其中包括若干 map 任务（map task）和若干 reduce 任务（reduce task），由 map 任务以完全并行的方式处理这些<key, value> 键值对，框架会对 map 的输出先进行排序，然后把结果输入给 reduce 任务。

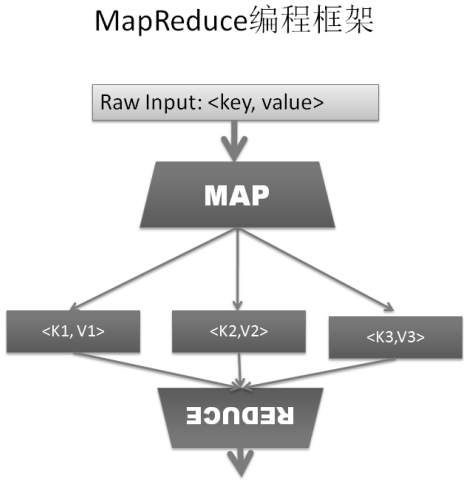


图 3-10 MapReduce 基本原理

Map 和 Reduce 这两个函数由程序员提供给系统，下层设施把 map 和 reduce 任务分布

在集群节点上运行，并把结果存储在 HDFS 上。整个 MapReduce 框架负责任务的调度和监控，以及重新执行已经失败的任务。

MapReduce 体系架构由一个单独的主 JobTracker 和每个集群节点一个从属的 TaskTracker 共同组成，如图 3-11 所示。

- JobTracker：负责协调和调度所有作业，并协调和调度构成一个作业的所有任务，这些任务分布在不同的 TaskTracker 上，JobTracker 监控它们的执行，重新执行已经失败的任务。如果一个任务失败了，JobTracker 可以将其重新部署到另一个 TaskTracker 上运行。
- TaskTracker：仅负责执行由 JobTracker 指派的任务，并向 JobTracker 发送进度报告（记录了每一个作业的运行进度）。

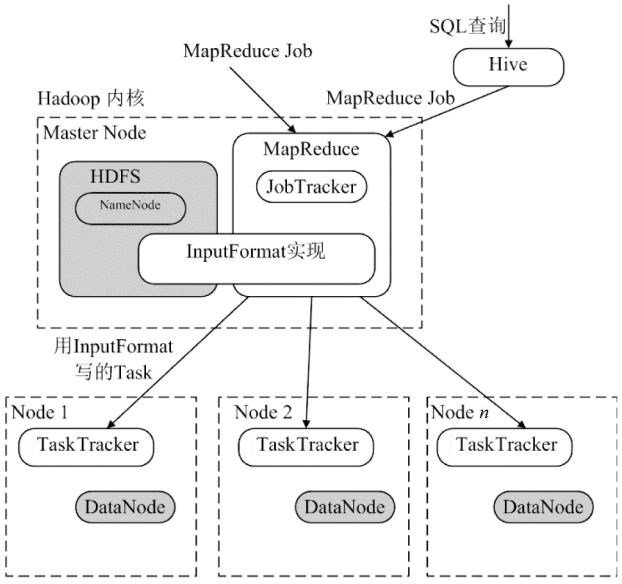


图 3-11　Hadoop 体系架构

通常，MapReduce 框架和 HDFS 是运行在一组相同的节点上的，即计算节点和存储节点通常在一起。这种配置允许 MapReduce 框架在那些已经存好数据的节点上高效地调度任务，这可以使整个集群的网络带宽并被非常高效地利用。因此，一个 Hadoop 的核心（Core），即一个主节点（Master Node），通常包含了 HDFS 的 NameNode 和 MapReduce 的 JobTracker，而一个 Hadoop 的节点（Node），通常包含了 HDFS 的 DataNode 和 MapReduce

的 TaskTracker，如图 3-11 所示。

在 Hadoop 中，文件是由一个 Hadoop Path 对象来表示的，可以把一个 Path 对象想象成一个 Hadoop 文件系统的 URI，例如 hdfs://localhost:9000/user/xt/input/text.dat。应用程序至少应该指明输入/输出路径，并通过实现合适的接口或抽象类提供 map 和 reduce 函数。再加上其他作业的参数，就构成了作业配置（job configuration）。然后，Hadoop 的作业客户端（job client）提交作业（jar 包/可执行程序等）和配置信息给 JobTracker，后者负责分发这些软件和配置信息给 TaskTracker、调度任务并监控它们的执行，同时提供状态和诊断信息给作业客户端。

虽然 Hadoop 框架是用 Java 实现的，但 MapReduce 应用程序则不一定要用 Java 来写。

- Hadoop Streaming 是一种运行作业的实用工具，它允许用户创建和运行任何可执行程序 （例如，Shell 工具）来作为 Mapper 和 Reducer。
- Hadoop Pipes 是一个与 SWIG 兼容的 C++ API （没有基于 JNI™ 技术），它也可用于实现 MapReduce 应用程序。

2. MapReduce 数据流

（1）一个 reduce 的数据流

MapReduce 的输入文件（Input file）被切分为固定大小的输入分块（Input splits，简称 split）。Hadoop 为每一个 split 都创建一个 map 任务，该 map 任务中的 map 函数会作用于 split 中的每一个记录（record）。一个记录就是一个<key, value>键值对。输入文件、输入分块和记录如图 3-12 所示。

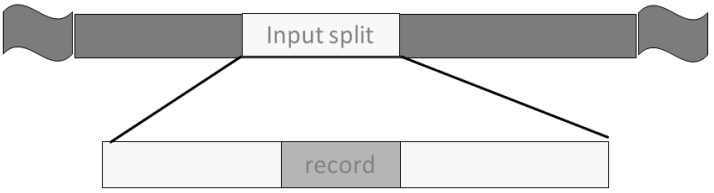


图 3-12　输入文件、输入分块和记录

Input split 是对 record（即<key, value>键值对）在逻辑上的切分，而 HDFS 中的块（Block）是对输入数据的物理切分。当两者一致时，很高效，但是实际中往往不一致。record 可能会跨越 HDFS 块的边界。因此，split 不该太大，过大可能失去并行性；也不能太小，

太小额外的开销占比会过大。与 HDFS 中的一个块的大小相同较为合适（块默认为 64MB）。

map 任务的输出将被写入磁盘（Linux 文件系统），而不是 HDFS 文件系统。因为，Map 的输出是中间临时结果（临时的<key, value>键值对），它们作为 reduce task 的输入。一旦作业结束，这些中间临时结果即被丢弃，不再需要。如果存入 HDFS，就需要复制多份副本在网络上传输，造成浪费。而 reduce 任务的输出会被写入到 HDFS 文件系统中，因为它们的输出是用户最终需要的结果，要妥善保存。一个 Reduce 的数据流图如图 3-13 所示。

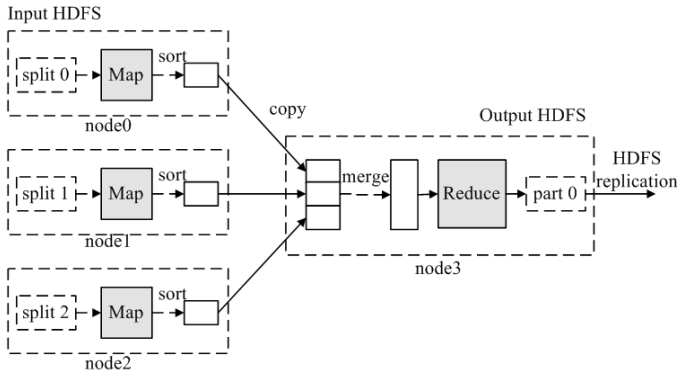


图 3-13　一个 reduce 的数据流

（2）多个 reduce 的数据流

当存在多个 Reducer 时，map 任务会将自己的输出进行分区（partition），然后再将其送至 reduce 任务。每一个 partition 会被发送至某个特别的 reduce 任务。一个 partition 内可以包含多个不同的 key，同一个 key 一定在一个 partition 内。多个 reduce 的数据流图如图 3-14 所示。

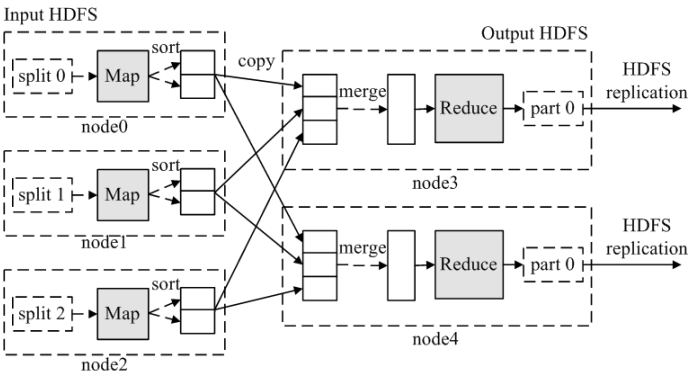


图 3-14　多个 reduce 的数据流

（3）没有 reduce 任务的数据流

可以只有 map 任务，而没有 reduce 任务，例如 Sqoop 作业。在这种情况下，map 任务的输出会被写入 HDFS。没有 reduce 任务的数据流图如图 3-15 所示。

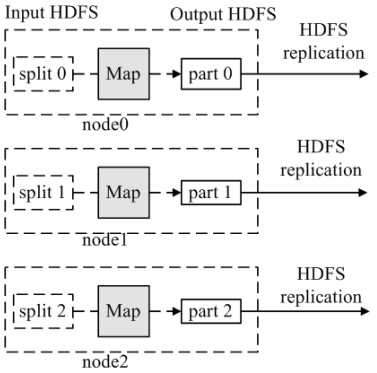


图 3-15　没有 reduce 任务的数据流

（4）shuffle 和 sort

MapReduce 保证每一个 Reducer 的输入都是按照 key 进行排序（sorted-by-key）的，在 Hadoop 中，按照 key 进行排序，并将 map 的输出传送至 reduce 的输入的过程，称为 shuffle（“洗牌”），详细分析如下文及图 3-16。

◎　map 端

在 map 端，每一个 map 任务都有循环缓存，默认大小为 100MB，map 任务将自己的输出写入该缓存，一个后台线程对缓存中的数据进行分区（partition）。分区的根据是这些数据将被送至哪一个 reduce 任务，因为每一个 partition 将被发送至一个 Reducer。在每一个 partition 内，线程将根据 key 对该 partition 内的数据（即<key, value>对）进行在内存中的排序。如果有 Combiner，Combiner 将作用于上述按 key 排序的输出。

当该缓存中写入的内容大小达到某个阈值（默认为 80%）时，一个后台线程开始将内容溢出（spill）到磁盘上。此时 map 任务写缓存的操作仍然继续，缓存如果满了，map task 将阻塞直到溢出完成。每当缓存的内容达到阈值时（后台线程开始溢出时），一个溢出文件（spill file）就会被创建。当 map 任务输出完最后一个 record 时，就可能存在多个溢出文件。在 map 任务完成之前，这些多个溢出文件将被归并（merge）到单独一个被分区和排序的输出文件（Output file）中。该输出文件在结构上由多个 partition 组成，每一个 partition

内的数据都是排好序的，且每一个 partition 将被送至对应的一个 reduce 任务。当指定了一个 Combiner 函数且溢出的数量达到 3 个时，那么在写输出文件之前，Combiner 会运行。

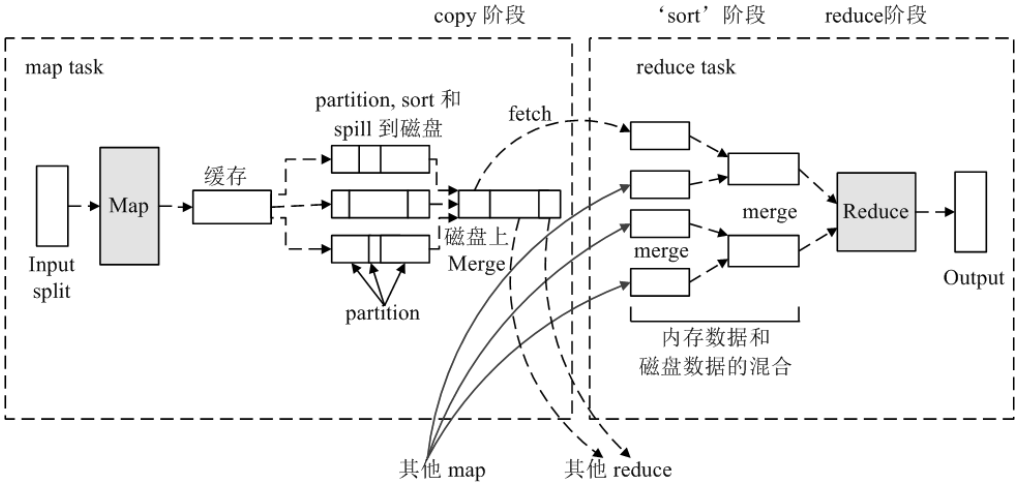


图 3-16　MapReduce 的 shuffle 和 sort 阶段

◎　reduce 端

当 map 任务成功完成后，它会将状态更新通知它所属的 TaskTracker，该 TaskTracker 进而又会通知其所属的 JobTracker。这些通知是通过心跳通信机制实现的。这样，对于一个作业而言，JobTracker 知道 map 输出与 TaskTracker 之间的映射关系。

多个 map 任务可能先后完成，任意一个 map 任务一旦完成，reduce 任务就开始复制它们的输出（特定的 partition），作为自己的输入，复制到自己所在的 HDFS 中，且一个 reduce 任务将会从多个 map 任务复制其所需要的 partition（这些 partition 都是同一类的）。reduce 任务取完自己的输入（即 map 任务的输出）后，这些数据并不是立即删掉，以应对 reduce 任务失败的情况，这称为复制阶段（copy）。Reducer 中的一个线程会周期性地向 JobTracker 询问 map 输出所在的位置，直到该 Reducer 接收了所有的 map 输出。当 reduce 任务复制了 map 任务的全部输出时，就开始将这些输出归并到少数几个文件（并保持文件内的 record 原来的排序），这称为归并阶段（merge）。

完整的 MapReduce 数据流如图 3-17 所示。

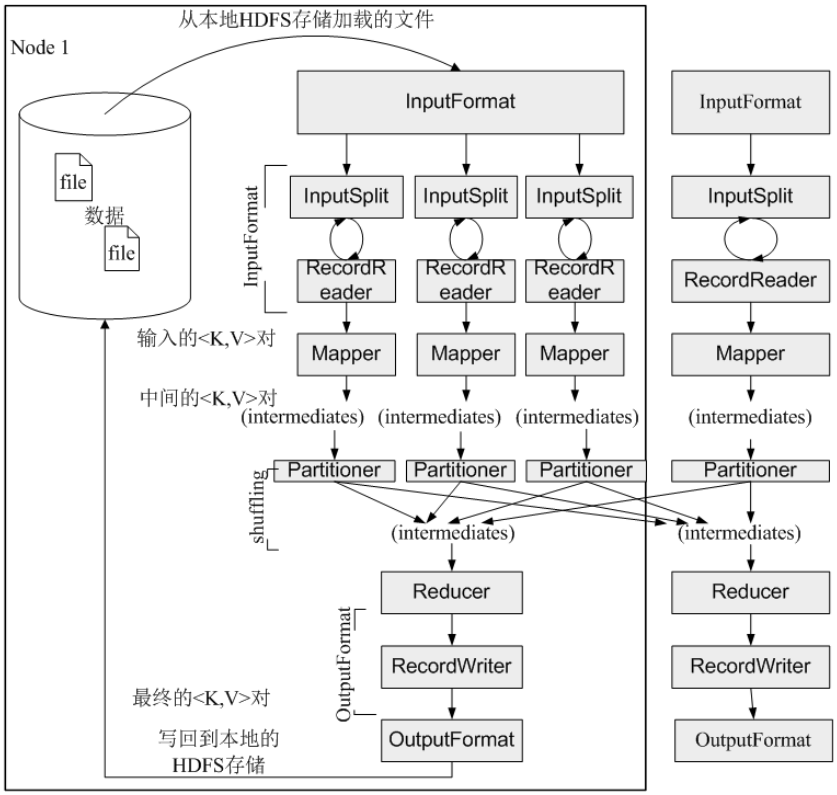


图 3-17　完整的 MapReduce 数据流

3. MapReduce 参数

Apache 的官网网站全面地介绍了 MapReduce 框架的各个方面[8]。应用程序通常会通过提供 map 和 reduce 来实现 Mapper 和 Reducer 接口，它们组成作业的核心。每个类/接口的 Javadoc 文档提供了最全面的文档；本书对这些参数的简介只是起到指南的作用。主要参数如表 3-6 所示。

[8] http://hadoop.apache.org/docs/stable/mapred_tutorial.html。

表 3-6　MapReduce 参数列表

参　数	作　用	默 认 值	其 他 实 现
InputFormat	将输入的数据集切割成小数据集 InputSplits，每一个 InputSplit 将由一个 Mapper 负责处理。InputFormat 中还提供一个 RecordReader 的实现，将一个 InputSplit 解析成 <key,value> 对提供给 map	TextInputFormat（针对文本文件，按行将文本文件切割成 InputSplits，并用 LineRecordReader 将 InputSplit 解析成<key,value>对，key 是行在文件中的位置，value 是文件中的一行）	SequenceFileInputFormat
OutputFormat	提供一个 RecordWriter 的实现，负责输出最终结果	TextOutputFormat（用 LineRecordWriter 将最终结果写成纯文件，每个<key,value>对一行，key 和 value 之间用 Tab 分隔）	SequenceFileOutputFormat
OutputKey Class	输出的最终结果中 key 的类型	LongWritable	
OutputValue Class	输出的最终结果中 value 的类型	Text	
Mapper Class	Mapper 类，实现 map 函数，完成输入的 <key,value> 到中间结果的映射	IdentityMapper（将输入的 <key,value> 原封不动地输出为中间结果）	LongSumReducer，LogRegexMapper，InverseMapper
Combiner Class	实现 combine 函数，将中间结果中的重复 key 做合并	Null（不对中间结果中的重复 key 做合并）	
Reducer Class	Reducer 类，实现 reduce 函数，对中间结果做合并，形成最终结果	IdentityReducer（将中间结果直接输出为最终结果）	AccumulatingReducer, LongSumReducer
InputPath	设定作业的输入目录，作业运行时会处理输入目录下的所有文件	null	
OutputPath	设定作业的输出目录，作业的最终结果会写入输出目录下	null	

续表

参　数	作　用	默 认 值	其 他 实 现
MapOutputKey Class	设定 map 函数输出的中间结果中 key 的类型	如果用户没有设定的话，使用 OutputKey Class	
MapOutputValue Class	设定 map 函数输出的中间结果中 value 的类型	如果用户没有设定的话，用 OutputValues Class	
OutputKeyComparator	对结果中的 key 进行排序时使用的比较器	Writable Comparable	
Partitioner Class	对中间结果的 key 排序后，用此 partition 函数将其划分为 *R* 份，每份由一个 Reducer 负责处理	HashPartitioner（使用 Hash 函数做 partition）	KeyFieldBasedPartitioner PipesPartitioner

下面对其中一些参数做简单的介绍，便于读者理解主要参数的应用。

（1）Mapper

MapReduce 框架为每一个 InputSplit 产生一个 map 任务。Mapper 将输入记录集（即<key, value>键值对）映射到一组中间格式的记录集（即一组键值对集合）。这种转换的中间格式记录集不需要与输入记录集的类型一致。一个给定的输入键值对可以映射成 0 个或多个输出键值对。

概括地说，对 Mapper 的实现首先需要重写 JobConfigurable.configure(JobConf)方法，这个方法需要传递一个 JobConf 参数，目的是完成 Mapper 的初始化工作。然后，框架为这个任务的 InputSplit 中每个键值对调用一次：map(WritableComparable,Writable, OutputCollector, Reporter)操作，其中：框架需要对 key 和 value 的类进行序列化操作，因此，这些类需要实现 Writable 接口；另外，为了方便框架执行排序的操作，key 类必须实现 WritableComparable 接口；通过调用 OutputCollector.collect(WritableComparable, Writable)可以收集输出的键值对；应用程序可以使用 Reporter 报告进度，设定应用级别的状态消息，更新 Counters（计数器），或者仅是表明自己运行正常。应用程序可以通过重写 Closeable.close()方法来执行相应的清理工作。

MapReduce 框架随后会把与一个特定 key 关联的所有中间过程的值（value）分成组，然后把它们传给 Reducer 以产出最终的结果。用户可以通过 JobConf.setOutputKeyComparatorClass(Class)

来指定具体负责分组的 Comparator。

Mapper 的输出被排序后，就被划分给每个 Reducer。分块的总数目和一个作业的 reduce 任务的数目是一样的。用户可以通过实现自定义的 Partitioner 来控制哪个 key 被分配给哪个 Reducer。

用户可选择通过 JobConf.setCombinerClass(Class)指定一个 Combiner，它负责对中间过程的输出进行本地的聚集，这会有助于降低从 Mapper 到 Reducer 数据传输量。

这些被排好序的中间过程的输出结果保存的格式是(key-len, key, value-len, value)，应用程序可以通过 JobConf 控制对这些中间结果是否进行压缩以及怎么压缩，使用哪种 CompressionCodec。

map 任务的数目通常是由输入数据的大小决定的，一般就是所有输入文件的总块（Block）数。map 任务正常的并行规模大致是每个节点（node）10～100 个 map 任务，对于 CPU 消耗较小的 map 任务可以设到 300 个左右。由于每个任务初始化需要一定的时间，因此，比较合理的情况是 map 任务执行的时间至少超过 1 分钟。这样，如果你输入 10TB 的数据，每个块的大小是 128MB，你将需要大约 82 000 个 map 任务来完成，除非使用 setNumMapTasks(int)将这个数值设置得更高。

（2）Reducer

Reducer 将与一个 key 关联的一组中间数值集归约（reduce）为一个更小的数值集。用户可以通过 JobConf.setNumReduceTasks(int)设定一个作业中 reduce 任务的数目。

概括地说，对 Reducer 的实现者需要重写 JobConfigurable.configure(JobConf)方法，这个方法需要传递一个 JobConf 参数，目的是完成 Reducer 的初始化工作。然后，框架为成组的输入数据中的每个<key, (list of values)>对调用一次 reduce(WritableComparable, Iterator, OutputCollector, Reporter)方法。之后，应用程序可以通过重写 Closeable.close()来执行相应的清理工作。

Reducer 有 3 个主要阶段：shuffle、sort 和 reduce。shuffle 和 sort 两个阶段是同时进行的。map 的输出也是一边被取回一边被合并的。

◎ shuffle

Reducer 的输入就是 Mapper 已经排好序的输出。在这个阶段，框架通过 HTTP 为每个 Reducer 获得所有 Mapper 输出中与之相关的分块。

◎ sort

这个阶段，框架将按照 key 的值对 Reducer 的输入进行排序（因为不同 Mapper 的输出中可能会有相同的 key）。如果需要中间过程对 key 的排序规则和 reduce 前对 key 的排序规则不同，那么可以通过 JobConf.setOutputValueGroupingComparator(Class)来指定一个 Comparator。再加上 JobConf.setOutputKeyComparatorClass(Class)可用于控制中间过程的 key 如何被分组，所以结合两者可以实现按值的二次排序。

◎ reduce

在这个阶段，框架为已分组的输入数据中的每个<key, (list of values)>对调用一次 reduce(WritableComparable, Iterator, OutputCollector, Reporter)方法。reduce 任务的输出通常是通过调用 OutputCollector.collect(WritableComparable, Writable)写入文件系统的。应用程序可以使用 Reporter 报告进度，设定应用程序级别的状态消息，更新 Counters（计数器），或者仅表明自己运行正常。Reducer 的输出是没有排序的。

reduce 任务的数目建议如下。

0.95 或 1.75 乘以 (<*no. of nodes*> * mapred.tasktracker.reduce.tasks.maximum)

其中：用 0.95，所有 reduce 任务可以在 map 任务一完成时就立刻启动，开始传输 Map 的输出结果。用 1.75，速度快的节点可以在完成第 1 轮 reduce 任务后，开始第 2 轮，这样可以得到比较好的负载均衡的效果。增加 reduce 任务的数目会增加整个框架的开销，但可以改善负载均衡，降低由于执行失败带来的负面影响。上述比例因子比整体数目稍小一些是为了给框架中的推测性任务（speculative-tasks）或失败的任务预留一些 reduce 的资源。

◎ 无 Reducer

如果没有归约要进行，那么设置 reduce 任务的数目为零是合法的。这种情况下，map 任务的输出会直接被写入由 setOutputPath(Path)指定的输出路径。框架在把它们写入 FileSystem 之前没有对它们进行排序。

（3）Partitioner

Partitioner 负责控制 map 任务输出结果 key 的分割。key（或者一个 key 子集）被用于产生分区，通常使用的是 Hash 函数。分区的数目与一个作业的 reduce 任务的数目是一样的。因此，它控制将中间过程的 key（也就是这条记录）应该发送给多个 reduce 任务中的

哪一个来进行 reduce 操作。HashPartitioner 是默认的 Partitioner。

（4）InputFormat：作业输入格式

每个 InputSplit 是由该作业的 InputFormat 产生的。InputFormat 为 MapReduce 作业描述输入的细节规范。MapReduce 框架根据作业的 InputFormat 来检查作业输入的有效性。基于文件的 InputFormat 实现（通常是 FileInputFormat 的子类） 默认行为是按照输入文件的字节大小，把输入文件切分成多个逻辑输入分块（InputSplit ）的实例，并把每一实例分别分发给一个 Mapper。 其中输入文件所在的文件系统（HDFS）的数据块尺寸是 InputSplit 大小的上限。下限可以设置 mapred.min.split.size 的值。InputFormat 中还提供 RecordReader 的实现，这个 RecordReader 从逻辑 InputSplit 中获得输入记录，这些记录将由 Mapper 处理。

考虑到边界情况，对于很多应用程序来说，按照文件大小进行逻辑输入分块是不能满足需求的。在这种情况下，应用程序需要实现一个 RecordReader 来处理记录的边界，并为每个任务提供一个逻辑输入分块的面向记录的视图。

TextInputFormat 是默认的 InputFormat。如果一个作业的 InputFormat 是 TextInputFormat，并且框架检测到输入文件的后缀是.gz 和.lzo，就会使用对应的 CompressionCodec 自动解压缩这些文件。但是需要注意，上述带后缀的压缩文件不会被切分，并且整个压缩文件会分给一个 Mapper 来处理。

（5）InputSplit

InputSplit 是一个单独的 Mapper 要处理的数据块。一般的 InputSplit 是字节样式输入，然后由 RecordReader 处理并转化成记录样式。FileSplit 是默认的 InputSplit，它把 map.input.file 设定为输入文件的路径，输入文件是逻辑分块文件。

（6）RecordReader：读取记录

RecordReader 从 InputSplit 读入<key, value>对。一般地，RecordReader 把由 InputSplit 提供的字节样式的输入文件转化成由 Mapper 处理的记录样式的文件。因此 RecordReader 负责处理记录的边界情况和把数据表示成<key,value>对形式。

（7）OutputFormat：作业输出

OutputFormat 描述 MapReduce 作业的输出样式。MapReduce 框架根据作业的 OutputFormat

来：检验作业的输出，例如检查输出路径是否已经存在。提供一个 RecordWriter 的实现，用来输出作业结果。输出文件保存在 FileSystem 上。TextOutputFormat 是默认的 OutputFormat。

（8）RecordWriter

RecordWriter 生成<key, value> 对到输出文件。RecordWriter 的实现把作业的输出结果写到 FileSystem。

（9）OutputCollector：收集数据

OutputCollector 是一个 MapReduce 框架提供的用于收集 Mapper 或 Reducer 输出数据的通用机制（包括中间输出结果和作业的输出结果）。

（10）Reporter：报告进度

Reporter 用于 MapReduce 应用程序报告进度，设定应用级别的状态消息，更新 Counters（计数器）的机制。Mapper 和 Reducer 的实现可以利用 Reporter 来报告进度，或者仅表明自己运行正常。在那种应用程序需要花很长时间处理个别键值对的场景中，这种机制是很关键的，因为框架可能会以为这个任务超时了，从而将它强行杀死。另一个避免这种情况发生的方式是，将配置参数 mapred.task.timeout 设置为一个足够高的值（或者干脆设置为零，则没有超时限制）。应用程序可以用 Reporter 来更新 Counter（计数器）。

（11）JobConf

一般情况，用户利用 JobConf 创建应用程序并配置作业属性，然后用 JobClient 提交作业并监视它的进程。

JobConf 代表一个 MapReduce 作业的配置。JobConf 是用户向 Hadoop 框架描述一个 MapReduce 作业如何执行的主要接口。

框架会按照 JobConf 描述的信息忠实地尝试完成这个作业，然而，一些参数可能会被标记为 final，这意味它们不能被更改；一些作业的参数可以被直接进行设置，例如，setNumReduceTasks(int)用于设置 reduce 的任务数；而另一些参数则与框架或者作业的其他参数之间有微妙地相互影响，设置起来比较复杂，例如对 map 任务数的设置 setNumMapTasks(int)。

通常，JobConf 会指明 Mapper、Combiner（如果有的话）、Partitioner、Reducer、InputFormat 和 OutputFormat 的具体实现。

JobConf 能指定一组输入文件（setInputPaths(JobConf, Path...) /addInputPath(JobConf, Path)）和（setInputPaths(JobConf, String) /addInputPaths(JobConf, String)）以及输出文件的位置（setOutputPath(Path)）。

JobConf 可有选择地对作业设置一些高级选项，例如：

- 设置 Comparator。
- 设置放到 DistributedCache 上的文件，DistributedCache 是面向大规模只读数据的。
- 设置中间结果或者作业输出结果是否需要压缩以及怎么压缩。
- 设置利用用户提供脚本(setMapDebugScript(String)/setReduceDebugScript(String))进行调试。
- 设置作业是否允许推测性（speculative）任务的执行（setMapSpeculativeExecution(boolean)）/（setReduceSpeculativeExecution(boolean)）。
- 设置每个任务最大的尝试次数（setMaxMapAttempts(int)/setMaxReduceAttempts(int)）。
- 设置一个作业能容忍的任务失败的百分比（setMaxMapTaskFailuresPercent(int)/setMaxReduceTaskFailuresPercent(int)）等。

（12）JobClient

JobClient 是用户提交的作业与 JobTracker 交互的主要接口。JobClient 提供提交作业、追踪进程、访问子任务的日志记录，获得 MapReduce 集群状态信息等功能。

作业提交过程包括：检查作业输入/输出样式细节，为作业计算 InputSplit 值。如果需要的话，为作业的 DistributedCache 建立必需的统计信息。复制作业的 jar 包和配置文件到 FileSystem 上的 MapReduce 系统目录下。提交作业到 JobTracker 并且监控它的状态。

将作业的历史文件记录到指定目录的“_logs/history/”子目录下。这个指定目录由 hadoop.job.history.user.location 设定，默认是作业输出的目录。因此默认情况下，文件会存放在 mapred.output.dir/_logs/history 目录下。用户可以设置 hadoop.job.history.user.location 为 none 来停止日志记录。

用户使用下面的命令可以看到在指定目录下的历史日志记录的摘要：

```
$ bin/hadoop job -history output-dir
```

这个命令会打印出作业的细节，以及失败的和被杀死的任务细节。要查看有关作业的更多细节，例如成功的任务、每个任务尝试的次数（task attempt）等，可以使用下面的命令：

```
$ bin/hadoop job -history all output-dir
```

用户可以使用 OutputLogFilter 从输出目录列表中筛选日志文件。

（13）任务的执行：mapred.child.java.opts

TaskTracker 是在一个单独的 JVM 上以子进程的形式执行 Mapper/Reducer 任务的。子任务会继承父 TaskTracker 的环境。mapred.child.java.opts 用于设置 TaskTracker 启动的子任务。

用户可以通过 JobConf 中的 mapred.child.java.opts 配置参数来设定子 JVM 上的附加选项，例如，通过-Djava.library.path=<> 将一个非标准路径设为运行时的链接，用以搜索共享库。如果 mapred.child.java.opts 包含一个符号*@taskid@*，它会被替换成 MapReduce 的 taskid 的值。下面是一个包含多个参数和替换的例子，其中包括：记录 JVM GC 日志；JVM JMX 代理程序以无密码的方式启动，这样它就能连接到 JConsole 上，从而可以查看子进程的内存和线程，得到线程 dump；还把子 JVM 的最大堆尺寸设置为 512MB，并为子 JVM 的 java.library.path 添加了一个附加路径。

```
<property>
  <name>mapred.child.java.opts</name>
  <value>
    -Xmx512M -Djava.library.path=/home/mycompany/lib -verbose:gc
    -Xloggc:/tmp/@taskid@.gc -Dcom.sun.management.jmxremote.authenticate=false
    -Dcom.sun.management.jmxremote.ssl=false
  </value>
</property>
```

用户或管理员也可以使用 mapred.child.ulimit 设定运行的子任务的最大虚拟内存。mapred.child.ulimit 的值以（KB）为单位，并且必须大于或等于-Xmx 参数传给 JVM 的值，否则 JVM 会无法启动。

（14）Task 的环境：mapred.local.dir

${mapred.local.dir}/taskTracker/，这是 TaskTracker 的本地目录，用于创建本地缓存和作业。它可以指定多个目录（跨越多个磁盘），文件会半随机地保存到本地路径下的某个目录。

当作业启动时，TaskTracker 根据配置文档创建本地作业目录，目录结构如以下所示。

① **${mapred.local.dir}/taskTracker/archive/**：分布式缓存，这个目录保存本地的分布式缓存。因此本地分布式缓存是在所有任务和作业间共享的。

② **${mapred.local.dir}/taskTracker/jobcache/$jobid/**：本地作业目录。

- **${mapred.local.dir}/taskTracker/jobcache/$jobid/work/**：作业指定的共享目录。各个任务可以使用这个空间作为暂存空间，用于它们之间共享文件。这个目录通过 job.local.dir 参数显示给用户。这个路径可以通过 JobConf.getJobLocalDir()来访问。它也可以作为系统属性获得。因此，用户（比如运行 streaming）可以调用 System.getProperty("job.local.dir")获得该目录。
- **${mapred.local.dir}/taskTracker/jobcache/$jobid/jars/**：存放 jar 包的路径。

  用于存放作业的 jar 文件和展开的 jar。job.jar 是应用程序的 jar 文件，它会被自动分发到各台机器，在任务启动前会被自动展开。使用 JobConf.getJar() 函数可以得到 job.jar 的位置。使用 JobConf.getJar().getParent()可以访问存放展开的 jar 包的目录。
- **${mapred.local.dir}/taskTracker/jobcache/$jobid/job.xml**：一个 job.xml 文件，本地的通用作业配置文件。
- **${mapred.local.dir}/taskTracker/jobcache/$jobid/$taskid**：每个任务有一个目录，它里面有如下的目录结构。
  - **${mapred.local.dir}/taskTracker/jobcache/$jobid/$taskid/job.xml**：一个 job.xml 文件，本地化的任务作业配置文件。任务本地化是指为该任务设定特定的属性值。下面的属性是为每个任务执行时使用的本地参数，它们保存在本地化的任务作业配置文件里，如表 3-7 所示。
  - **${mapred.local.dir}/taskTracker/jobcache/$jobid/$taskid/output**：一个存放中间过程的输出文件的目录。它保存了由框架产生的临时 map 和 reduce 数据，比如 map 的输出文件等。
  - **${mapred.local.dir}/taskTracker/jobcache/$jobid/$taskid/work**：任务的当前工作目录。

- **${mapred.local.dir}/taskTracker/jobcache/$jobid/$taskid/work/tmp：**任务的临时目录。（用户可以设定属性 mapred.child.tmp 来为 map 和 reduce task 设定临时目录。默认值是./tmp。如果这个值不是绝对路径，它会把任务的工作路径加到该路径前面作为任务的临时文件路径。如果这个值是绝对路径则直接使用这个值。如果指定的目录不存在，会自动创建该目录。之后，按照选项 -Djava.io.tmpdir='临时文件的绝对路径'执行 Java 子任务。Pipes 和 Streaming 的临时文件路径是通过环境变量 TMPDIR='the absolute path of the tmp dir'设定的）。如果 mapred.child.tmp 有./tmp 值，这个目录会被创建。

表 3-7　本地任务作业配置文件

名　称	类　型	描　述
mapred.job.id	String	作业 id
mapred.jar	String	作业目录下 job.jar 的位置
job.local.dir	String	作业指定的共享存储空间
mapred.tip.id	String	任务 id
mapred.task.id	String	任务尝试 id
mapred.task.is.map	boolean	是否是 map 任务
mapred.task.partition	int	任务在作业中的 id
map.input.file	String	map 读取的文件名
map.input.start	long	map 输入的数据块的起始位置偏移
map.input.length	long	map 输入的数据块的字节数
mapred.work.output.dir	String	任务临时输出目录

任务的标准输出和错误输出流会被读到 TaskTracker 中，并且记录到：

```
${HADOOP_LOG_DIR}/userlogs
```

DistributedCache 可用于 map 或 reduce 任务中分发 jar 包和本地库。子 JVM 总是把当前工作目录加到 java.library.path 和 LD_LIBRARY_PATH。因此，可以通过 System.loadLibrary 或 System.load 装载缓存的库。

（15）任务的 Side-Effect File

在一些应用程序中，子任务需要产生一些 side-file，这些文件与作业实际输出结果的文件不同。在这种情况下，同一个 Mapper 或者 Reducer 的两个实例（比如推测性任务）

同时打开或者写 FileSystem 上的同一文件就会产生冲突。因此应用程序在写文件的时候需要为每次任务尝试（不仅仅是每次任务，每个任务可以尝试执行很多次）选取一个独一无二的文件名（使用 attemptid，例如 task_200709221812_0001_m_000000_0）。

为了避免冲突，MapReduce 框架为每次尝试执行任务都建立和维护一个特殊的${mapred.output.dir}/_temporary/_${taskid}子目录，这个目录位于本次尝试执行任务输出结果所在的 FileSystem 上，可以通过 ${mapred.work.output.dir}来访问这个子目录。对于成功完成的任务尝试，只有${mapred.output.dir}/_temporary/_${taskid}下的文件会移动到${mapred.output.dir}。当然，框架会丢弃那些失败的任务尝试的子目录。这种处理过程对于应用程序来说是完全透明的。

在任务执行期间，应用程序在写文件时可以利用这个特性，比如通过 FileOutputFormat. getWorkOutputPath()获得${mapred.work.output.dir}目录，并在其下创建任意任务执行时所需的 side-file，框架在任务尝试成功时会马上移动这些文件，因此不需要在程序内为每次任务尝试选取一个独一无二的名字。

注意：在每次任务尝试执行期间，${mapred.work.output.dir} 的值实际上是 ${mapred.output.dir}/_temporary/_{$taskid}，这个值是 MapReduce 框架创建的。所以使用这个特性的方法是，在 FileOutputFormat.getWorkOutputPath() 路径下创建 side-file 即可。对于只使用 map 不使用 reduce 的作业，这个结论也成立。这种情况下，map 的输出结果直接生成到 HDFS 上。

（16）其他有用的特性

◎ Counters

Counters 是多个由 MapReduce 框架或者应用程序定义的全局计数器。每一个 Counter 可以是任何一种 Enum 类型。同一特定 Enum 类型的 Counter 可以汇集到一个组，其类型为 Counters.Group。应用程序可以定义任意（Enum 类型）的 Counters 并且可以通过 map 或者 reduce 方法中的 Reporter.incrCounter（Enum, long）或者 Reporter.incrCounter(String, String, long)更新，之后 MapReduce 框架会汇总这些全局 Counters。

◎ DistributedCache

DistributedCache 可将具体应用相关的、大尺寸的、只读的文件有效地分布放置。DistributedCache 是 MapReduce 框架提供的功能，能够缓存应用程序所需的文件（包括文

本、档案文件、jar 文件等）。应用程序在 JobConf 中通过 URL（hdfs://）指定需要被缓存的文件。DistributedCache 假定由 hdfs://格式 URL 指定的文件已经在 FileSystem 上了。

MapRedcue 框架在作业所有任务执行之前会把必要的文件复制到从属节点上。它运行高效是因为每个作业的文件只复制一次并且为那些没有文档的从属节点缓存文档。DistributedCache 根据缓存文档修改的时间戳进行追踪。在作业执行期间，当前应用程序或者外部程序不能修改缓存文件。

DistributedCache 可以分发简单的只读数据或文本文件，也可以分发复杂类型的文件，例如归档文件和 jar 文件。归档文件（zip、tar、tgz 和 tar.gz 文件）在从属节点上会被解档（un-archived）。这些文件可以设置执行权限。

用户可以通过设置 mapred.cache.{files|archives}来分发文件。如果要分发多个文件，可以使用逗号分隔文件所在路径，也可以利用 API 来设置该属性：DistributedCache.addCacheFile (URI,conf)/DistributedCache.addCacheArchive(URI,conf)和 DistributedCache.setCacheFiles(URIs, conf)/DistributedCache.setCacheArchives(URIs,conf)，其中 URI 的形式是 hdfs://host:port/absolute-path#link-name。在 Streaming 程序中，可以通过命令行选项-cacheFile/-cacheArchive 分发文件。

用户可以通过 DistributedCache.createSymlink(Configuration)方法让 DistributedCache 在当前工作目录下创建到缓存文件的符号链接，或者通过设置配置文件属性 mapred.create.symlink 为 yes。分布式缓存会截取 URI 的片段作为链接的名字。例如，URI 是 hdfs://namenode:port/lib.so.1#lib.so，则在任务当前工作目录会有名为 lib.so 的链接，它会链接分布式缓存中的 lib.so.1。

DistributedCache 可在 MapRedcue 任务中作为一种基础软件分发机制使用。它可以被用于分发 jar 包和本地库（native libraries）。DistributedCache.addArchiveToClassPath(Path, Configuration)和 DistributedCache.addFileToClassPath(Path, Configuration) API 能够被用于缓存文件和 jar 包，并把它们加入子 JVM 的 classpath，也可以通过设置配置文档里的属性 mapred.job.classpath.{files|archives}达到相同的效果。缓存文件可用于分发和装载本地库。

◎ Tool

Tool 接口支持处理常用的 Hadoop 命令行选项，Hadoop 命令行的常用选项有：

```
-conf <configuration file>
-D <property=value>
```

```
-fs <local|namenode:port>
-jt <local|jobtracker:port>
```

◎ IsolationRunner

IsolationRunner 是帮助调试 MapReduce 程序的工具。使用 IsolationRunner 的方法是，首先设置 keep.failed.tasks.files 属性为 true（同时参考 keep.tasks.files.pattern）。然后，登录到任务运行失败的节点上，进入 TaskTracker 的本地路径运行 IsolationRunner。

```
$ cd <local path>/taskTracker/${taskid}/work
$ bin/hadoop org.apache.hadoop.mapred.IsolationRunner ../job.xml
```

IsolationRunner 会把失败的任务放在单独的一个能够调试的 JVM 上运行，并且采用和之前完全一样的输入数据。

◎ Profiling

Profiling 是一个工具，它使用内置的 Java profiler 工具进行分析，获得 2、3 个 map 或 reduce 样例运行分析报告。

用户可以通过设置属性 mapred.task.profile 指定系统是否采集 profiler 信息。利用 JobConf.setProfileEnabled(boolean)可以修改属性值。如果设为 true，则开启 profiling 功能。profiler 信息保存在用户日志目录下。默认情况，profiling 功能是关闭的。

用户可以使用配置文档里的属性 mapred.task.profile.{maps|reduces}设置要 profile map/reduce 任务的范围。设置该属性值的 API 是 JobConf.setProfileTaskRange(boolean, String)。范围的默认值是 0～2。

用户可以设定配置文档里的属性 mapred.task.profile.params 来指定 profiler 配置参数。改属性要使用 JobConf.setProfileParams(String)。当运行任务时，如果字符串包含%s。它会被替换成 profiling 的输出文件名。这些参数会在命令行里传递到子 JVM 中。默认的 profiling 参数是：-agentlib:hprof=cpu=samples,heap=sites,force=n,thread=y,verbose=n,file=%s。

◎ 调试

MapReduce 框架能够运行用户提供的用于调试的脚本程序。当 MapReduce 任务失败时，用户可以通过运行脚本在任务日志（例如任务的标准输出、标准错误、系统日志以及作业配置文件）上做后续处理工作。用户提供的调试脚本程序的标准输出和标准错误会输出为诊断文件。如果需要的话，这些输出结果也可以打印在用户界面上。

为了提交调试脚本，首先要把这个脚本分发出去，而且还要在配置文件里设置。用户要用 DistributedCache 机制来分发和链接脚本文件。一个快速提交调试脚本的方法是分别为需要调试的 map 任务和 reduce 任务设置"mapred.map.task.debug.script"和"mapred.reduce.task.debug.script"属性的值。这些属性也可以通过 JobConf.setMapDebugScript (String)和 JobConf.setReduceDebugScript(String)来设置。对于 Streaming，可以分别为需要调试的 map 任务和 reduce 任务使用命令行选项-mapdebug 和-reducedegug 来提交调试脚本。脚本的参数是任务的标准输出、标准错误、系统日志以及作业配置文件。在运行 map/reduce 失败的节点上运行调试命令是：

```
$script $stdout $stderr $syslog $jobconf
```

Pipes 程序根据第 5 个参数获得 C++程序名。因此调试 Pipes 程序的命令是：

```
$script $stdout $stderr $syslog $jobconf $program
```

◎ JobControl

JobControl 是一个工具，它封装了一组 MapReduce 作业以及它们之间的依赖关系。

◎ 数据压缩

Hadoop MapReduce 框架为应用程序的写入文件操作提供压缩工具，这些工具可以为 map 输出的中间数据和作业最终输出数据（例如 reduce 的输出）提供支持。它还附带了一些 CompressionCodec 的实现，比如实现了 zlib 和 lzo 压缩算法。Hadoop 同样支持 gzip 文件格式。考虑到性能问题（zlib）以及 Java 类库的缺失（lzo）等因素，Hadoop 也为上述压缩解压算法提供本地库的实现。

◎ 中间输出的压缩

应用程序可以通过 JobConf.setCompressMapOutput(boolean)来控制 map 输出的中间结果，并且可以通过 JobConf.setMapOutputCompressorClass(Class)来指定 CompressionCodec。

◎ 作业输出的压缩

应用程序可以通过：

```
FileOutputFormat.setCompressOutput(JobConf, boolean)
```

来控制输出是否需要压缩，并且可以使用：

```
FileOutputFormat.setOutputCompressorClass(JobConf, Class)
```

指定 CompressionCodec。如果作业输出要保存成 SequenceFileOutputFormat 格式，需要使用 SequenceFileOutputFormat.setOutputCompressionType(JobConf,SequenceFile.CompressionType) 来设定 SequenceFile.CompressionType，默认是 RECORD 类型。

4. 一个 MapReduce 的例子

Hadoop 提供了一个 WordCount 的例子，可以作为初学者学习 MapReduce 的开始，其流程如图 3-18 所示。

源代码 WordCount.java

```
1.  package org.myorg;
2.  import java.io.IOException;
3.  import java.util.*;
4.  import org.apache.hadoop.fs.Path;
5.  import org.apache.hadoop.conf.*;
6.  import org.apache.hadoop.io.*;
7.  import org.apache.hadoop.mapred.*;
8.  import org.apache.hadoop.util.*;
9.  public class WordCount {
10. public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
11. private final static IntWritable one = new IntWritable(1);
12. private Text word = new Text();
13. public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
14. String line = value.toString();
15. StringTokenizer tokenizer = new StringTokenizer(line);
16. while (tokenizer.hasMoreTokens()) {
17. word.set(tokenizer.nextToken());
18. output.collect(word, one);
19. }
20. }
```

```
21. }
22. public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
23. public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
24. int sum = 0;
25. while (values.hasNext()) {
26 sum += values.next().get();
27. }
28. output.collect(key, new IntWritable(sum));
29. }
30. }
31. public static void main(String[] args) throws Exception {
32. JobConf conf = new JobConf(WordCount.class);
33. conf.setJobName("wordcount");
34. conf.setOutputKeyClass(Text.class);
35. conf.setOutputValueClass(IntWritable.class);
36. conf.setMapperClass(Map.class);
37. conf.setCombinerClass(Reduce.class);
38. conf.setReducerClass(Reduce.class);
39. conf.setInputFormat(TextInputFormat.class);
40. conf.setOutputFormat(TextOutputFormat.class);
41. FileInputFormat.setInputPaths(conf, new Path(args[0]));
42. FileOutputFormat.setOutputPath(conf, new Path(args[1]));
43. JobClient.runJob(conf);
44. }
45. }
```

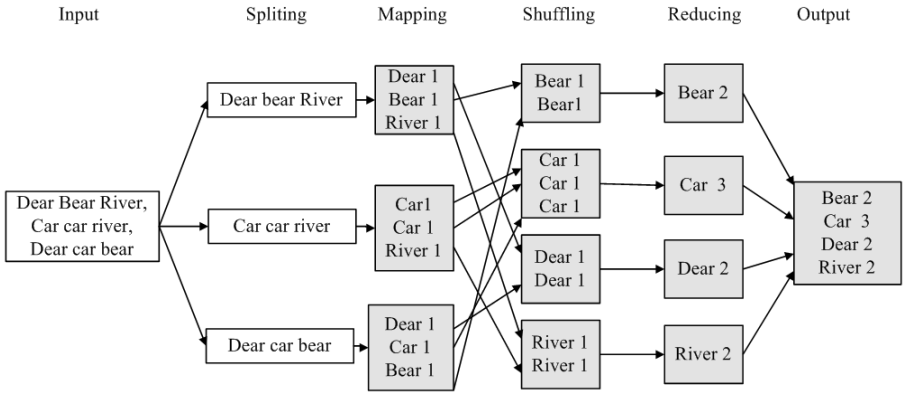


图 3-18　MapReduce 词汇记数的流程

5. MapReduce 集成开发环境

目前比较常用的 Java 集成开发环境（IDE）主要是 Eclipse，它是一个界面友好的开源 IDE，并支持成千上万种不同的插件，为代码分析和源码调试提供了很大的便利。可以在 Eclipse 官方网站找到 Eclipse 的各个版本[9]。

当前 Hadoop 版本自带有 Eclipse 插件，可以在安装目录下找到插件，例如对 Hadoop 0.20.2 版本，在 home/hadoop/hadoop-0.20.2/contrib/eclipse-plugin 下就有插件 hadoop-0.20.2-eclipse-plugin.jar，将这个 jar 包复制到 Eclipse 安装目录下的 plugins 里，然后打开 Eclipse，单击主菜单上的 window-preferences，在左边栏中找到 Hadoop Map/Reduce，单击后在右边对话框里设置 hadoop 的安装路径，即主目录。

插件安装完成后，就可以开始创建一个 MapReduce Project 了，单击 Eclipse 主菜单上的 File→New→Project，在弹出的对话框中选择 MapReduce Project，之后输入 Project 的名字，例如 wordcount，单击“确定”按钮即可。接着，就可以像任何一个普通的 Eclipse Java Project 一样，添加 Java 类。例如，可以定义一个 WordCount 类，然后将安装的 Hadoop 程序里的 WordCount 源程序代码写到此类中，就可在 Eclipse 中形成一个完整的 wordcount 程序。

[9] http://www.eclipse.org/downloads/。

6. YARN：MapReduce 2.0

MapReduce 在 Hadoop 0.23 经历了一次大规模更新，新版本的 MapReduce 2.0 被称为 YARN 或 MRv2。YARN 的框架如图 3-19 所示。

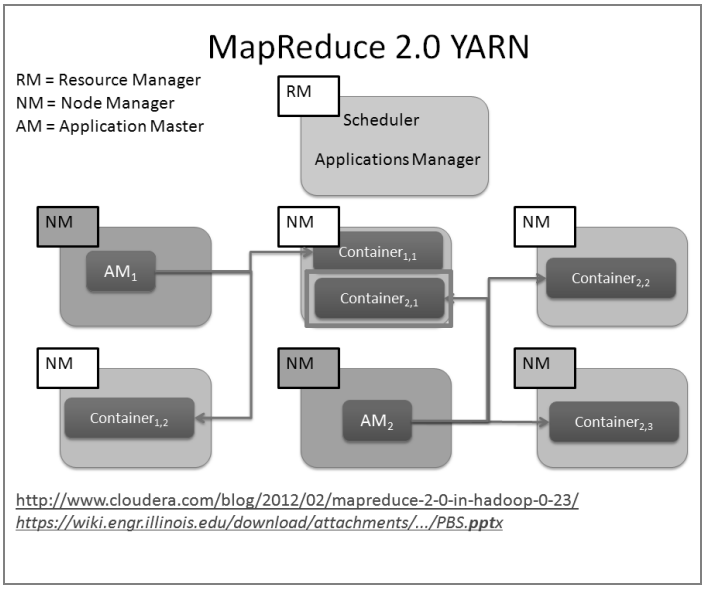


图 3-19　YARN 的框架

YARN 的基本思想是将 JobTracker 的两个主要功能（资源管理和作业调度/监控）分离，主要方法是创建一个全局的 ResourceManager（RM）和若干个针对应用程序的 ApplicationMaster（AM）。这里的应用程序是指传统的 MapReduce 作业或作业的 DAG（有向无环图）。ResourceManager（RM）和每个从属节点的 NodeManager（NM）构成了数据计算框架。

（1）资源管理器 ResourceManager（RM）

ResourceManager 从全局角度，负责将计算机资源分配到各个应用程序。ResourceManager 中有两个主要组件：Scheduler 和 ApplicationsManager。

- 调度器 Scheduler 负责给应用程序分配资源。Scheduler 从某种意义上说是一种纯粹的调度，它不监控和跟踪应用程序的状态，另外它也不负责重启应用程序或者硬件故障造成的失败。Scheduler 根据应用程序的资源需求执行调度，这些需求基

于一个抽象的资源概念：容器 Container，包括了内存、CPU、硬盘和网络等资源。每个客户机/应用可以请求多个资源。

- 应用管理器 ApplicationsManager 负责接收作业提交，将应用程序分配给具体的 ApplicationMaster，并负责重启失败的 ApplicationMaster。

（2）NodeManager（NM）

每个从属节点 NodeManager 是每台机器的框架代理，负责管理 Container，监控它们的资源使用情况（包括 CPU、内存、硬盘、网络），同时向 ResourceManager/Scheduler 汇报。

（3）ApplicationMaster（AM）

针对各个应用程序的 ApplicationMaster 管理了应用的生命周期（调度和协调）。ApplicationMaster 实际上是一个详细的框架库，它结合从 ResourceManager 获得的资源和 NodeManager 协同工作来运行和监控任务。ApplicationMaster 负责向 Scheduler 请求适当的资源 Container，启动任务，跟踪它们的使用状态并监控其进展。它也负责处理任务失败。一个应用程序或者是 MapReduce 作业中的一个单一作业或者是这些作业的一个 DAG。

YARN 在接口上兼容于此前的稳定版本（Hadoop 0.20.205），这意味着以前的 MapReduce 作业重新编译后就可以在 YARN 下运行。YARN 本质上是 Hadoop 的操作系统，突破了 MapReduce 框架的性能瓶颈[10]，构成了 Hadoop 2.0 生态系统的核心，如图 3-20 所示。

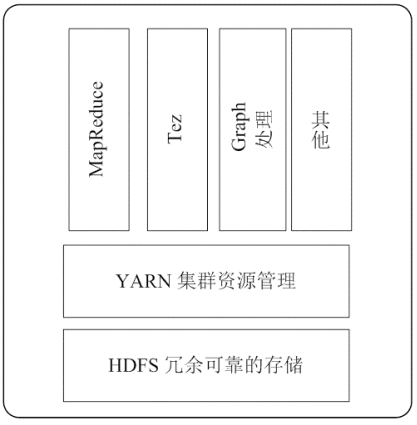


图 3-20　YARN 的生态系统

[10] expect-a-ripping-good-yarn-at-hadoop-summit，http://siliconangle.com/blog/2013/06/25/expect-a-ripping-good-yarn-at-hadoop-summit/。

YARN 在扩展性方面有很大改进，可以支持 10 000+计算机集群，改进了 MapReduce 的性能，并且支持 MapReduce 之外的其他计算框架，如低延时、流计算和服务等框架。

3.2.4 Hadoop 之 NoSQL：分布式数据库

1. NoSQL 数据库概述

大数据通常采用分布式存储，非关系型分布式数据库（NoSQL）是分布式存储的主要技术。一般有 4 种非关系型数据库管理系统，即基于列存储的 NoSQL、基于键值对的 NoSQL、图表数据库和基于文档的数据库。这些非关系型数据库管理系统将源数据聚集在一起，同时用 MapReduce 的分析程序来对汇总的信息进行分析。

（1）大数据对传统数据库提出的挑战

传统关系型数据库面临的挑战主要有以下几点，其需求、问题、解决思路如表 3-8 所示。

◎ 数据库高并发读写的挑战

大数据处理要求高并发读写，能高并发、实时动态地获取和更新数据。但目前的 RDBMS 存在的问题是数据库读写压力巨大，硬盘 I/O 无法承受。例如，在广泛应用的社交媒体网站中，要根据用户个性化信息来实时生成动态页面和提供动态信息，无法使用动态页面静态化技术，因此数据库的并发负载非常高，往往要达到每秒上万次的读写请求。此时的磁盘 I/O 根本无法承受如此之多的读写请求。为解决这一矛盾，RDBMS 提出的解决方案是主从分离（Master-Slave）；分库、分表，缓解写压力，增强读库的可扩展性等措施。

◎ 海量数据的高效率存储和访问的挑战

大数据要求支持海量数据的高效率存储和访问。但目前的 RDBMS 存在的问题是存储记录数量有限、SQL 查询效率极低。例如，在 Facebook、Twitter 等 SNS 网站中，每天用户产生海量的用户动态，以 Twitter 为例，一个月就达到了 2.5 亿条用户动态。要对海量用户信息进行高效率实时存储和查询，对于关系型数据库来说，在一张 2.5 亿条记录的表里面进行 SQL 查询，效率是极其低下的。再如大型 Web 网站的用户登录系统，如腾讯、盛大，数以亿计的账号，关系型数据库也难以应付。为解决这一矛盾，RDBMS 提出的解决方案是：分库、分表，缓解数据增长压力。

◎ 数据库的高扩展性和高可用性的挑战

大数据要求需要拥有快速横向扩展能力、提供 7×24 小时不间断服务。但目前的 RDBMS 存在的问题是横向扩展艰难，无法通过快速增加服务器节点实现；系统升级和维护造成服务不可用。在基于 Web 的架构中，数据库是最难进行横向扩展的，当用户量和访问量增加时，数据库没有办法像 Web 服务器那样简单地通过添加更多的硬件和服务节点来扩展性能和负载能力，对于很多需要 24 小时不间断服务的网站来说，对数据库系统的升级和扩展往往需要停机维护。为解决这一矛盾，RDBMS 提出的解决方案是：主从分离，增强读库的可扩展性；MySQL 的主数据复制管理（MMM）等。

表 3-8　传统关系型数据库面对大数据的挑战

需　求	问　题	解 决 思 路
对数据库高并发读写	数据库读写压力巨大，硬盘 I/O 无法承受	主从分离； 分库、分表，缓解写压力，增强读库的可扩展性
对海量数据的高效率存储和访问	存储记录数量有限、SQL 查询效率极低	分库、分表，缓解数据增长压力
对数据库的高扩展性和 高可用性	横向扩展艰难，无法通过快速增加服务器节点实现；系统升级和维护造成服务不可用	主从分离，增强读库的可扩展性； MySQL 的主数据复制管理（MMM）

但是，关系型数据库所采取的措施存在着明显的缺陷：分库、分表的缺点受业务规则影响，需求变动导致分库分表的维护十分复杂；系统数据访问层代码需要修改。主从分离（Master-Slave）的缺点是 Slave 实时性的保障，对于实时性很高的场合可能需要做一些处理；高可用性问题也受到挑战，Master 容易产生单点故障。MMM 的缺点是本身扩展性差，一次只能一个 Master 写入，只能解决有限数据量下的可用性。在改进这些缺点的背景下，NoSQL 数据库应运而生。

（2）CAP 原理

分布式数据库系统有一个著名的经证明的 CAP 原理，它是由 Eric Brewer 教授提出，经 Seth Gilbert 和 Nancy lynch 两人证明了 CAP 理论的正确性[11]。CAP 理论指出分布式数据系统有 3 个基本要素，即：

- 一致性（Consistency）。

[11] CAP 的参考资料：Nancy Lynch and Seth Gilbert, “Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services”, ACM SIGACT News, Volume 33 Issue 2 (2002), pg. 51-59。

- 可用性（Availability）。
- 分区容忍性（Partition tolerance）。

在分布式系统中，这 3 个要素最多只能同时实现两点，不可能三者兼顾，如图 3-21 所示。

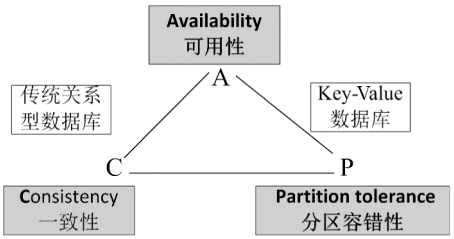


图 3-21　CAP 原理示意图

对于分布式数据库系统，分区容忍性是基本要求，对于大多数 Web 应用，牺牲一致性而换取高可用性（AP），是目前多数分布式数据库产品的主要方向。而传统的关系型数据库则主要追求可用性和一致性（CA）。

（3）NoSQL 数据库的概念和特性

NoSQL 是 Not Only SQL 的缩写，而不是 Not SQL，它不一定遵循传统数据库的一些基本要求，比如遵循 SQL 标准、ACID 属性、表结构等。相比传统数据库，叫它分布式数据管理系统更贴切，数据存储被简化且更灵活，重点被放在了分布式数据管理上。

NoSQL 数据库的主要特性包括如下。

- 易扩展

NoSQL 数据库种类繁多，但有一个共同的特点是去掉关系型数据库的关系型特性。数据之间无关系，这样就非常容易扩展。无形之间，在架构的层面上带来了可扩展的能力。甚至有多种 NoSQL 之间的整合。

- 灵活的数据模型

NoSQL 无须预先为要存储的数据建立字段，随时可以存储自定义的数据格式。而在关系型数据库里，增删字段是一件非常麻烦的事情。如果是非常大数据量的表，增加字段十分困难。

- 高可用

NoSQL 在不太影响性能的情况，可以方便地实现高可用的架构，比如通过复制模型实现高可用。

◎　大数据量，高性能

NoSQL 数据库都具有非常高的读写性能，尤其在大数据量下，表现优秀。这得益于它的无关系性，数据库的结构简单。

（4）NoSQL 数据库的两个核心理论基础

Google 的 BigTable。BigTable 提出了一种很有趣的数据模型，它将各列数据进行排序存储。数据值按范围分布在多台机器，数据更新操作有严格的一致性保证。

Amazon 的 Dynamo。Dynamo 使用的是另外一种分布式模型。Dynamo 的模型更简单，它将数据按 key 进行哈希存储。其数据分片模型有比较强的容灾性，因此它实现的是相对松散的弱一致性：最终一致性。

这两种系统不但已经开始商用，而且都公开了比较详细的实现论文[12、13]。它们各自实现架构迥异，存储特性不一，但都结构优美，技术上各有千秋，却又殊途同归。两者都是以（key,value）形式进行存储的，但 Dynamo 存储的数据是非结构化数据，对 value 的解析完全是用户程序的事情，Dynamo 系统不识别任何结构数据，都统一按照二进制数据对待；而 BigTable 存储的是结构化或半结构化数据——就如关系型数据库中的列一般，因而可支持一定程度的查询。对于架构而言，Dynamo 让数据在环中均匀“存储”，各存储点相互能通信，不需要 Master 主控点控制，优点是无单点故障危险，且负载均衡。BigTable 由一个主控服务器加上多个子表服务器构成，Master 主控服务器负责监控各客户存储节点，好处是更人为可控，方便维护，且集中管理时数据同步易于方便。

（5）NoSQL 数据库与关系型数据库的比较

大数据给传统的数据管理方式带来了严峻的挑战，关系型数据库在容量、性能、成本等多方面都难以满足大数据管理的需求。NoSQL 数据库通过折中关系型数据库严格的数据一致性管理，在可扩展性、模型灵活性、经济性和访问性等方面获得了很大的优势，可以更好地适应大数据应用的需求，成为大数据时代最重要的数据管理技术。两者比较如图 3-22 所示。

[12] http://research.google.com/archive/bigtable.html。

[13] http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html。

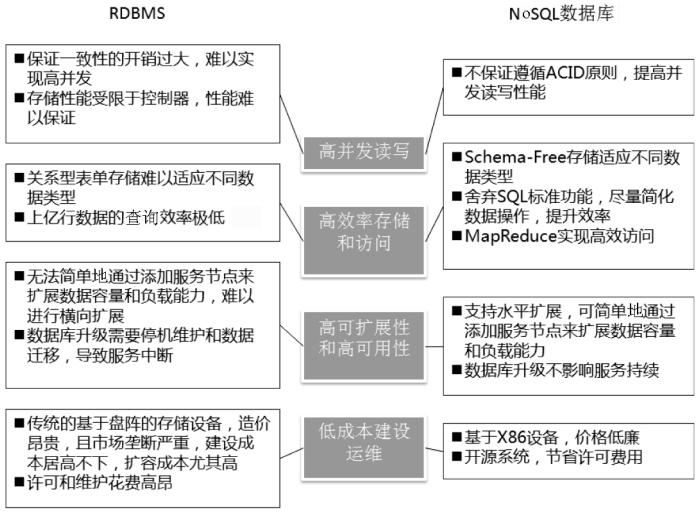


图 3-22　NoSQL 数据库与关系型数据库的比较

（6）主流的 NoSQL 数据库比较

NoSQL 数据库有很多种，大致分为 6 类，如表 3-9 所示为主流 NoSQL 数据库的比较。

表 3-9　主流 NoSQL 数据库的比较

类　型	部分代表	特　点	典型应用
列存储	BigTable HBase Cassandra Hypertable	按列存储数据，特点是方便存储结构化和半结构化数据，方便做数据压缩，对针对某一列或者某几列的查询有非常大的 I/O 优势。查找速度快、可扩展性强、更容易进行分布式扩展	汇总统计和数据仓库
key-value 存储	Tokyo Cabinet / Tyrant Berkeley DB Memcache DB Redis Dynamo Voldemort Oracle Coherence	可以通过 key 快速查询到其 value，查询速度很快。一般来说，存储不管 value 的格式，都会存下	大数据高负载应用、日志
文档存储 / 全文索引	MongoDB CouchDB	文档存储一般用类似 JSON 的格式存储，存储的内容是文档型的。有机会对某些字段建立索引，实现关系型数据库的某些功能。数据结构要求不严格，表结构可变	半结构和非结构数据存储

续表

类　型	部 分 代 表	特　点	典 型 应 用
图存储	Neo4J FlockDB InfoGrid HyperGraghDB、Infinite Gragh	高效匹配图结构相关算法，图形关系的最佳存储	社交网络、 推荐系统
对象存储	db4o Versant	通过类似面向对象语言的语法操作数据库，通过对象的方式存取数据	
XML 数据库	Berkeley DB XML BaseX	高效地存储 XML 数据，并支持 XML 的内部查询语法，比如 Xquery、Xpath	

2. HBase

HBase 是 Apache 在谷歌 BigTable 启发下开发的，HBase 和 BigTable 的基本原理是一致的，下面先简单介绍 BigTable，在此基础上，详细分析 HBase 的应用。

（1）源于 Google BigTable

BigTable 是 Google 的分布式结构化数据的存储系统，它被设计用来处理海量数据，可以是分布在数千台普通服务器上的 PB 级的数据。自 2005 年起，BigTable 在超过 60 个 Google 的产品和项目上得到了应用，包括 Google Analytics、Google Finance、Orkut、Personalized Search、Writely 和 Google Earth。它适用性很广泛、具有可扩展、高性能和高可用性，能执行高吞吐量的结构化数据的批处理，处理结果能及时地响应并快速返回给最终用户。

BigTable 是列存储数据库。行和列的值的数据类型都是 string 类型，因此 key-value 映射如下：(row:string, column:string, time:int64)→ string，其中，row、column 的值都为 string 类型，time 为 64 位整型。BigTable 通过行关键字的字典顺序来组织数据，对同一个行关键字的读或者写操作都是原子性的（不管读或者写这一行里多少个不同的列），而列是可以动态添加的。一个列中，不同版本的数据通过时间戳来索引，可以由用户程序赋值或 BigTable 生成。

BigTable 表中的每个行都可以根据行关键字动态分区。每个分区叫作一个片（Tablet），片是 BigTable 数据结构的基本单元，是负载均衡的单元。最初表都只有一个片，但随着表不断增大，片会自动分裂，片的大小控制在 100MB～200MB。BigTable 列关键字组成的集

合叫作“列族”，列族是访问控制的基本单位。访问控制、磁盘和内存的使用统计都是在列族层面进行的。

BigTable API 分为数据 API 和客户端 API，数据 API 包括：创建/删除表和列族，修改集群、表和列族的元数据。客户端操作 API 包括写/删除数据值、读数据值、行查找、一个行关键字下的数据进行原子性的读—更新—写操作，作为 MapReduce 框架的输入和输出。

BigTable 用 GFS 来存储日志和数据文件，按 SSTable（SSTable：SortedStringTable）文件格式存储数据。SSTable 是 key-value 映射的，建有块索引。打开 SSTable 的时候，索引被加载到内存，自动复制整个 SSTable 数据到内存中。BigTable 用 Chubby 管理元数据，Chubby 文件保存根 Tablet 的位置。BigTable 使用一个类似 B+树的数据结构存储片的位置信息。首先是第 1 层，Chubby file。它保存着根 Tablet 的位置；第 2 层是根 Tablet。根 Tablet 其实是元数据表（METADATA Table）的第 1 个分片，它保存着元数据表其他片的位置。根 Tablet 很特别，为了保证树的深度不变，根 Tablet 从不分裂。第 3 层是其他的元数据片，它们和根 Tablet 一起组成完整的元数据表。每个元数据片都包含了许多用户 Tablets 的位置信息。可以看出整个定位系统其实只是两部分，一个 Chubby 文件，另一个元数据表，如图 3-23 所示为 BigTable 的结构。

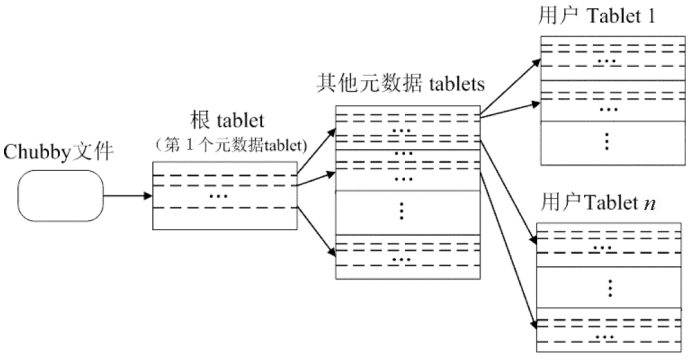


图 3-23　BigTable 的结构

BigTable 集群是由一个供客户端使用的库和一个主服务器（Master Server）、许多片服务器（Tablet Server）组成的。每个片服务器负责一定量的片，处理对其片的读写请求，以及片的分裂或合并。片服务器可以根据负载随时添加和删除。这里片服务器并不真实存储数据，而相当于一个连接 BigTable 和 GFS 的代理，客户端的一些数据操作都通过片服务器代理间接访问 GFS。主服务器负责将片分配给片服务器，监控片服务器的添加和删除，

平衡片服务器的负载，处理表和列族的创建等。主服务器不存储任何片，不提供任何数据服务，也不提供片的定位信息。

（2）Hadoop HBase 的特性

Apache HBase 是一个架构 Apache Hadoop 上的开源的、分布式的、可横向扩充的、一致的、低时延的、随机访问的非关系型数据库[14]。它是由谷歌 BigTable 的启发所立的项目，和 BigTable 如出一辙。HBase 的性能优势如下。

◎ HBase 横向扩展能力强。

增加更多的服务器能线性增加 HBase 性能和容量，包括存储容量和输入/输出的操作性能。HBase 最大能支持 1000 个节点，1PB 的集群。通常使用的 HBase 集群是 10～40 个节点，100GB～4TB 的容量。

◎ HBase 是遵循一致性的，它符合 CAP 原理。

- 一致性：它具有数据库风格的 ACID，在行的一致性上有保证。
- 可用性：在返回旧数据的基础上支持从错误中的及时恢复。
- 分区容错：如果一个节点崩溃，系统持续运行。

◎ HBase 提供低时延的随机访问。

- HBase 写操作：1～3 毫秒，每个节点每秒 1000～10 000 个写操作。
- HBase 读操作：内存读 0～3 毫秒，硬盘读 10～30 毫秒，从内存读每个节点每秒 10 000～40 000 个读操作。
- HBase 每个（row,column）称为一个单元（cell），单元尺寸为 0MB～3MB 较好。
- 在表的任何位置都可以读、写或者插入数据。
- 没有顺序写的限制。

表 3-10 是对 HBase 和关系型数据库管理系统（RDBMS）的比较。

表 3-10　RDBMS 和 HBase 的详细比较

	RDBMS	HBase
数据布局	面向行	面向列族
事务	多行 ACID	只是单一行

[14] Using HBase effectively - What You Need to Know as an Application Developer Presentation.pdf。

续表

	RDBMS	HBase
查询语言	SQL	get/put/scan/……
安全	认证和授权	认证、列限定词级授权
索引	在任意列	只在行关键字
最大数据尺寸	TB 级	1PB
读写并发最大限制	每秒 1000 次查询	每秒成百上千万次查询

（3）HBase 的数据类型

图 3-24 是一个 HBase 的表，每行有一个主关键字，行由列组成，每个(row,column)即一个单元（cell）。单元的内容都是 byte[]类型。应用必须知道类型，并处理它们。Byte[]按照字典顺序排列，时间戳与数据的多个版本相联系，行是强一致性的。

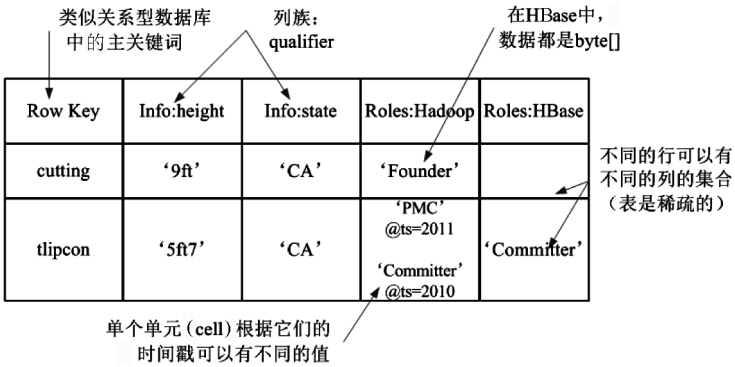


图 3-24　HBase 表结构

列族（Column Family）是一组相关的列，如图 3-24 所示的第 2、3 列是 Info 列族（Info 称为列族限定词），第 4 和 5 列是 Roles 列族，这组列的集合有相似的访问模式。列族的物理存储特点是每个列族被包含在它自己的文件中。在磁盘上，列族按照行关键字、列关键字和降序的时间戳排序。如表 3-11 所示为列族的示例。

表 3-11　列族的示例

Info 列族

Row Key	Column Key	Timestamp	Cell Value
cutting	Info:height	1273516197868	9ft

续表

Row Key	Column Key	Timestamp	Cell Value
cutting	Info:state	1043871824184	CA
tlipcom	Info:height	1273878447049	5ft7
tlipcon	Info:state	1273616297446	CA

roles 列族

Row Key	Column Key	Timestamp	Cell Value
cutting	Roles:ASF	1273871823022	Director
cutting	Roles:Hadoop	1183746289103	Founder
tlipcom	Roles: Hadoop	1300062064923	PMC
tlipcon	Roles: Hadoop	1293388212294	Committer
tlipcon	Roles:Hive	1273616297446	Contributor

列族里面，选择参数可以来调整每个列族的读操作的性能，若把相关数据存储在一起，可以更好地压缩。列族的不同参数包括块压缩（none、gzip、LZO、Snappy）、版本保留策略和内存优先级等。选择不同的参数，可以获得不同的性能。

表被分成很多行的集合，称为区（Region，即 HRegion）。横向扩展的读和写的能力是通过跨许多 Region 来扩展实现的。区类似于 BigTable 的片。每个 HRegion 由多个 Store 构成，每个 Store 由一个 MemStore 和 0 个或多个 StoreFile 组成。每个 Store 保存一个列族，StoreFile 以 HFile 格式存储在 HDFS 中。

空表提供了 Schema 的灵活性，以后增加列不必转变整个 Schema。因此，HBase 被称为“以查询为中心的 Schema 设计”。这与关系型数据库“以 Schema 为中心的设计”不同。

（4）HBase 的体系架构

HBase 是列式数据库，架构在 Hadoop HDFS 上，采用分布式计算架构 MapReduce，所用语言是 Java，协议是 HTTP/REST，能支持数十亿行与上百万列的存储。HBase 集群包括 HMaster，它与 HDFS NameNode 部署在一起；Region 服务器，它们与 DataNode 部署在一起，调度由 ZooKeeper 来协调。Hbase 的体系架构如图 3-25 所示。

HMaster 负责控制哪一个 Region 被哪个 Region 服务器来服务。当 Region 新到来或者失效时，安排 Region 到一个新的 Region 服务器。原来的 Master 如果停机，备用 Master

变成活的 master。这些转换被 ZooKeeper 来协调。Apache ZooKeeper 是专职协调的高可用系统，通常有 3～5 个机器（总是奇数），ZooKeeper 用这个机器的一致意见来保证公共共享的状态。

Region 服务器负责服务 Region。一个 Region 一次只被单独的一个 Region 服务器服务，Region 服务器可以服务多个 Region。如果 Region 服务器停机，实现自动的负载均衡。Region 服务器（RS）与 DataNode（DN）在同样的位置，充分利用 HDFS 文件的本地性。

列族操作在 MemStore（在内存，按照 map 排序）上执行，包含最后修改的行。MemStore 溢出后在硬件数据文件上生成 HFile。

HBase 写操作的路径是通过 Put 执行数据输入，在 Region 的 MemStore 上进行操作，内存写满后生成 HFile。

生成的多个 HFile 可以进行合并，实现 Region 的 HFile 压缩；一个 Region 也可以进行分割形成几个 Region；不同 Region 之间可以进行负载的均衡。

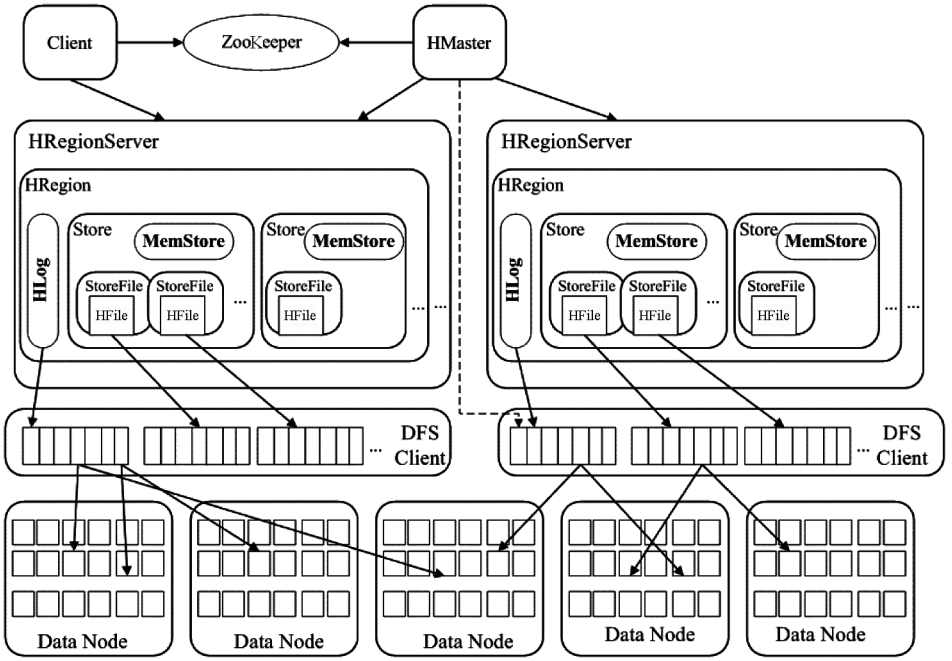


图 3-25　HBase 体系架构

（5）访问 HBase

Java 客户端 API 可以实现对 HBase 的访问，包括数据操作 API 和 DDL 操作。数据操作包括的 API 有：

- Get，输出数据，get(byte [] row, byte [] column, long timestamp, int versions)。
- Put，输入数据。
- Scan，扫描数据。
- CheckAndPut，输入前检查并输入数据。
- Delete，删除数据。

DDL 操作包括的 API 有：

- Create，生成。
- Alter，变更。
- Enable/Disable，使能或取消。

数据访问也可以通过非 Java 客户端 API 的其他工具和接口，包括：

- 非 Java 的客户端。
  - Thrift 服务器支持 HBase Client 的实例。
- Ruby、C++和通过 Thrift 的 Java 客户端。
  - REST 服务器支持 HBase Client 的访问。
- MapReduce 的 TableInput/OutputFormat。
  - HBase 作为 MapReduce（source）或者接收地（sink）。
- HBase Shell。
  - JRuby IRB 与“DSL”来增加 get、scan 和 admin 等。
  - *./bin/hbase shell YOUR_SCRIPT*。

HBase 可以对实时查询进行优化，借助高性能 Thrift/REST 网关，通过在 Server 端扫描及过滤实现对查询操作预判，支持 XML、Protobuf 和 Binary 的 HTTP，基于 JRuby（JIRB）的 Shell。

Thrift 是一个 Facebook 提供的软件框架，它用于可扩展的跨语言的服务开发，能够在 C++、Java、Python、PHP 和 Ruby 等语言之间实现无缝的支持。这将启动 Thrift 服务器的实例，默认在 9090 端口。类似的项目是 REST。

REST 表述性状态转移（REpresentational State Transfer，REST），作为一种设计 Web 服务的方法，它定义了如何正确使用 Web 的标准，包括 5 个原则：一是为所有“事物”定义 ID，二是将所有事物连接在一起，三是使用标准方法，四是资源的多重表述，五是无状态通信。通过基于 REST 的 API 公开系统资源是一种灵活的方法，可以为不同种类的应用程序提供以标准方式格式化的数据。如图 3-26 所示为 HBase 的应用体系架构。

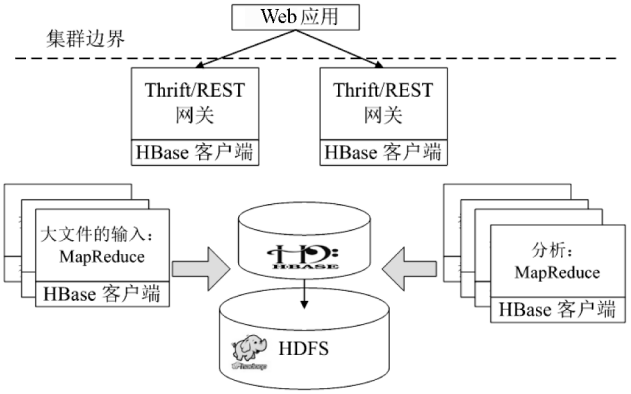


图 3-26　HBase 的应用体系架构

在 Hadoop 的框架下，Apache Hadoop HDFS 用于数据持久性和可靠性（写之前的日志），Apache ZooKeeper 用于分布式协调，Apache Hadoop MapReduce 支持内置的运行 MapReduce 作业。HBase 分析能力不强，不支持 SQL 语言，可以与 Hive、Sqoop 实现集成。如表 3-12 所示为 HDFS/MapReduce 与 HBase 的比较。

表 3-12　HDFS/MR 与 HBase 的比较

	HDFS/MapReduce	HBase
抽象	文件+字节	表+行
写的模式	只是附加	随机写和大批量增加
读的模式	全文件扫描、分区表扫描（Hive）	随机读，小范围扫描或表扫描
结构化存储	自己存、TSV、顺序文件、Avro	空列族数据模型
最大数据尺寸	30PB 以上	1PB

3. 其他典型的 NoSQL

（1）Cassandra

Apache Cassandra 是一套开源分布式 NoSQL 数据库系统。它最初由 Facebook 开发，用于储存收件箱等简单格式数据，集 Google BigTable 的数据模型和 Amazon Dynamo 的完全分布式的架构于一身。Facebook 于 2008 将 Cassandra 开源，此后，由于 Cassandra 良好的可扩展性，被 Digg、Twitter 等知名 Web 2.0 网站所采纳，成为了一种流行的分布式结构化数据存储方案。

- 使用许可：Apache。
- 所用语言：Java。
- 协议：Custom、Binary。
- 特点：数据类型是类似 BigTable 的列族，在数据库中增加一列非常方便，写操作比读操作更快；体系架构又类似 Dynamo，是基于分布式哈希表的完全对等连接（P2P）架构，与传统的基于共享的数据库集群相比，Cassandra 可以几乎无缝地加入或删除节点。
- 适合应用场景：写操作远多过读操作（如日志记录），节点规模变化比较快的应用场景，每个系统组建都必须用 Java 编写的场景。

（2）CouchDB

CouchDB 是一个开源的面向文档的数据库管理系统，可以通过符合 REST 标准的 JSON API 访问。

- 使用许可：Apache。
- 所用语言：Erlang。Erlang 是一种通用的面向大规模并发活动的编程语言。
- 协议：HTTP/REST。
- 特点：遵循数据库一致性，易于使用。
- 适用应用场景：很适合 CMS、电话本、地址簿等应用。

（3）MongoDB

MongoDB 是一个基于分布式文件存储的数据库，旨在为 Web 应用提供可扩展的高性

能数据存储解决方案。MongoDB 是一个介于关系型数据库和非关系型数据库之间的产品，是非关系型数据库当中功能丰富且又类似关系型数据库的产品。

- 使用许可：AGPL（发起者：Apache）。
- 所用语言：C++。
- 协议：Custom、Binary（BSON）。
- 特点：支持的数据结构非常松散，是类似 JSON 的 BSON 格式，因此可以存储比较复杂的数据类型。支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，可以实现类似关系型数据库单表查询的绝大部分功能，还支持对数据建立索引。支持 Javascript 表达式查询。
- 最佳应用场景：适用于需要动态查询支持，需要使用索引的分布式应用，需要对大数据库有性能要求，需要使用 CouchDB，但因为数据改变太频繁而占满内存的应用程序。目前优酷的在线评论业务已部分迁移到 MongoDB。

（4）Redis

Redis 是一个高性能的 key-value 存储系统。

- 使用许可：BSD。
- 所用语言：C/C++。
- 协议：类 Telnet。
- 特点：一是支持存储的 value 类型相对很多，包括 string（字符串）、list（链表）、set（集合）和 zset（有序集合）。这些数据类型都支持 push/pop、add/remove 及取交集、并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，Redis 支持各种不同方式的排序。二是 Redis 运行异常快，为了保证效率，数据都是缓存在内存中的。三是虽然采用简单数据或以关键字索引的哈希表，但也支持复杂操作。四是 Redis 支持事务，支持将数据设置成过期数据。
- 最佳应用场景：适用于数据变化快且数据库大小可预见（适合内存容量）的应用程序。例如，股票价格、数据分析、实时数据收集、实时通信。目前新浪微博是 Redis 全球最大的用户，在新浪有 200 多台物理机、400 多个端口正在运行着 Redis，有+4GB 的数据跑在 Redis 上来为微博用户提供服务。

（5）MemBase

- 使用许可：Apache 2.0
- 所用语言：Erlang 和 C。
- 协议：分布式缓存及扩展。
- 特点：兼容 Memcache，但同时兼具持久化和支持集群。非常快速（200kb+/秒），通过关键字索引数据；可持久化存储到硬盘；所有节点都是唯一的（Master-Master 复制）；在内存中同样支持类似分布式缓存的缓存单元；写数据时通过去除重复数据来减少 I/O ；提供非常好的集群管理 Web 界面；更新软件时无须停止数据库服务；支持连接池和多路复用的连接代理。
- 最佳应用场景：适用于需要低时延数据访问、支持高并发以及高可用性的应用程序。例如，低时延数据访问，比如以广告为目标的应用；高并发的 Web 应用，比如网络游戏（Zynga）。

（6）Neo4j

- 使用许可：GPL，其中一些特性使用 AGPL/商业许可。
- 所用语言：Java。
- 协议：HTTP/REST（或嵌入在 Java 中）。
- 特点：基于关系的图形数据库，可独立使用或嵌入到 Java 应用程序，图形的节点和边都可以带有元数据，很好地自带 Web 管理功能，使用多种算法支持路径搜索，使用关键字和关系进行索引，为读操作进行优化，支持事务（用 Java API），使用 Gremlin 图形遍历语言，支持 Groovy 脚本，支持在线备份、高级监控及高可靠性，支持使用 AGPL/商业许可。
- 最佳应用场景：适用于图形一类数据。这是 Neo4j 与其他 NoSQL 数据库的最显著区别，例如，社会关系、公共交通网络、地图及网络拓扑。

3.2.5　Hadoop 之外的大数据计算技术

图 3-27 形象地表示了大数据处理不同技术及其相互关系。实时的大数据流中非结构

化文档在文件层面都是存储在 HDFS 中的，数据的处理方式有 4 种：一是执行 Hadoop MapReduce 的批处理；二是在实时 NoSQL 数据库，如 HBase 中进行处理；三是经商业化较成熟的 BigSQL 进行处理；四是实时流数据的实时处理，如 S4、Storm 进行处理。表 3-13 是大数据不同存储和处理方式的比较。

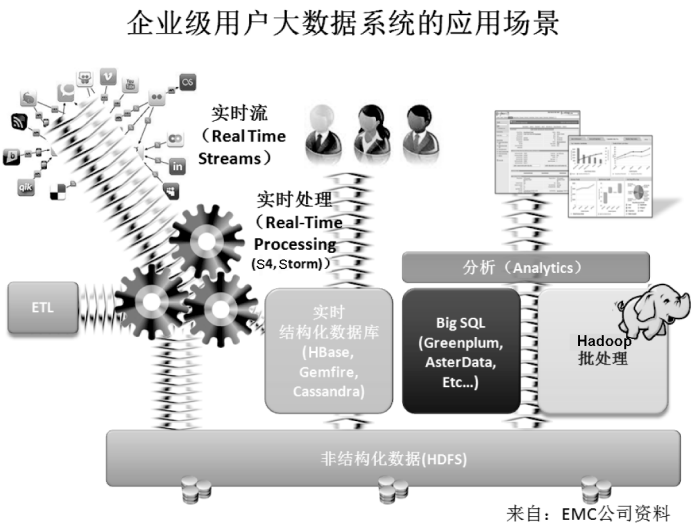


图 3-27　大数据处理的不同方式[15]

表 3-13　大数据不同存储和处理方式的比较

	分布式文件系统	大规模 NoSQL	内存计算	BigSQL
数据类型	日志文件、机器产生的数据、设备数据等	松散类型设备数据、记录、事件、统计数、复杂关系/图谱	结构化的、可分区的数据	结构化数据
数据规模	100 PB	数万亿行	十亿行	PB
集群规模	1000	100	10～100	100

[15] http://cdn.oreillystatic.com/Architecting Virtualized Infrastructure for Big Data Presentation.ppt。

续表

	分布式文件系统	大规模 NoSQL	内存计算	BigSQL
技术	NAS GFS HDFS Blob（S3、Atmos 等）	BigTable Cassandra HBase Voldemort	Redis Gemfire Membase	Netezza Greenplum Sybase IQ AsterData 等
价值	存储任何数据， 易于扩张， 能优化成本	易于扩张， 灵活和动态的 Schema's	高并发 低延迟	对重复查询的高性能， 查询语言易用性
数据可计算	是	将来	Hybrid Possible	不行
本地磁盘	是 付出成本和带宽	是 付出成本和可用性	主要是内存	是 付出成本和带宽

1. BigSQL

BigSQL 是一种分布式数据技术，它基于大规模并行处理系统（MPP）。它可以把关系型数据库架构在 Hadoop 上，从而实现 SQL+ MapReduce，从而可以用关系型数据库来替代 HBase。BigSQL 设计初衷是面向大规模数据分析的，能轻松扩展到 PB 级别。通过 BigSQL 的并行数据流引擎，能够让程序员编 MapReduce，数据库管理员（DBA）继续做 SQL，似乎两者优势兼得。EMC 公司的 Greenplum、Teradata 公司的 AsterData 等都是这方面的代表。BigSQL 特性如下。

（1）基于大规模并行处理系统（MPP）的技术

BigSQL 一般采用 MPP 架构。在 MPP 系统中，每个 SMP 节点也可以运行自己的操作系统、数据库等，这意味着每个节点内的 CPU 不能访问另一个节点的内存。节点之间的信息交互是通过节点互联网络实现的，这个过程一般称为数据重分布（Data Redistribution）。与传统的 SMP 架构明显不同，通常情况下，MPP 系统因为要在不同处理单元之间传送信息，所以它的效率要比 SMP 差一点。但是 MPP 系统不共享资源，因此对它而言资源比 SMP 多。在 OLTP 程序中，用户访问一个中心数据库，如果采用 SMP 系统结构，它的效率要比采用 MPP 结构快得多；而 MPP 系统在决策支持和数据挖掘方面显示了优势。

BigSQL 采用的 MPP 架构，其主要优点是大规模的并行处理能力，并行数据流引擎是其核心，如图 3-28 所示。

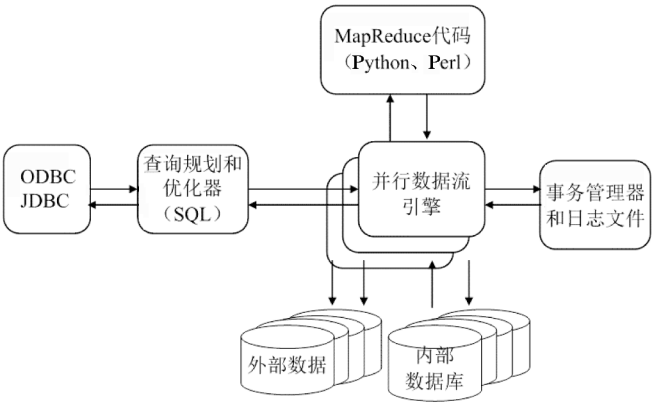


图 3-28　BigSQL 的工作原理

（2）采用无共享架构

常见的在线事务处理（OLTP）数据库系统常采用 shared everything 架构来做集群，例如 Oracle RAC 架构（如图 3-29 所示），数据存储共享，节点间内存可以相互访问。

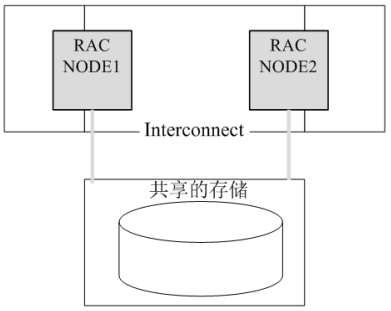


图 3-29　Oracle RAC 架构

BigSQL 是一种分布式数据库。其采用无共享（shared nothing）架构，主机、操作系统、内存、存储都是自我控制的，不存在共享。其架构主要由 Master Host（控制节点）、Segment Host（数据节点）和 Interconnect 三大部分组成。整个集群由很多个 Segment Host、Master Host 组成，其中每个 Segment Host 上运行了很多个开源的关系型数据库，如图 3-30 所示。

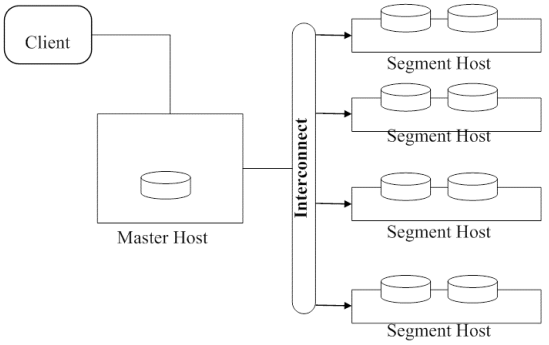


图 3-30　BigSQL 的架构图（以 Greenplum 为例）

Master 节点的主要作用是接收客户端的连接、处理 SQL 命令、调配各 Segment 节点间的工作负载、协调各 Segment 节点返回结果并把最终的结果返回给用户。所有数据库的元数据都保存在 Master 节点，并不保存用户数据。各 Segment 数据要做交换的是不经过 Master 的。

Segment 节点的主要作用是数据存储、处理大多数的查询请求。表和索引被分布在数据库的可用 Segment 节点中，每个 Segment 包含部分且唯一的数据。用户不能直接和 Segment 节点做交互，都要先通过 Master 节点。

Interconnect 网络连接层的作用是负责各 Segment 节点进程通信，使用标准的千兆交换机。数据传输默认使用 UDP 协议。使用 UPD 时，数据库会做额外数据包校验和，对未执行的也会做检查。故在可靠性上，基本和 TCP 是等价的，在性能和扩展性上，却优于 TCP。使用 TCP 的话，数据库有 1000 个 Segment 的限制，UDP 则没有。

BigSQL 分布式数据库通过将数据分布到多个节点上来实现规模数据的存储。每个表都是分布在所有节点上的，Master Host 首先通过对表的某个或多个列进行哈希运算，然后根据运算结果将表的数据分布到 Segment Host 中。整个过程中 Master Host 不存放任何用户数据，只有对客户端进行访问控制和存储表分布逻辑的元数据。数据被规律地分布到节点上，充分利用 Segment 主机的 I/O 能力，以此让系统达到最大的 I/O 能力。

分布式数据库的并行处理主要体现在外部表并行装载、并行备份恢复与并行查询处理 3 个方面。数据仓库的主要精力一般集中在数据的装载和查询，数据的并行装载主要是在采用外部表或者 Web 表方式，通常情况下通过 gpfdist 来实现。gpfdist 程序能够以 370MB/s 的速度装载文本文件和以 200MB/s 的速度装载 CSV 格式文件，ETL 带宽为 1GB 的情况下，可

以运行 3 个 gpfdist 程序装载文本文件，或者运行 5 个 gpfdist 程序装载 CSV 格式文件。图 3-31 中采用了两个 gpfdist 程序进行数据装载。可以根据实际的环境通过配置 postgresql.conf 参数文件来优化装载性能。

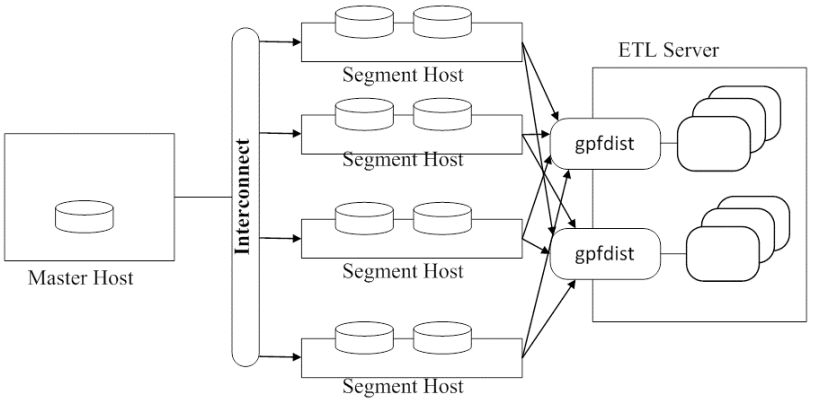


图 3-31　并行装载外部文件实例（以 Greenplum 为例）

并行查询性能的强弱往往由查询优化器的水平来决定，主节点负责解析 SQL 与生成执行计划。在 Master Host 存在 Query Dispatcher（QD）进程，该进程前期负责查询计划的创建和调度，Segment Instance 返回结果后，该进程再进行聚合与向用户展示；Segment Host 存在 Query Executor（QE）进程，该进程负责其他节点相互通信与执行 QD 调度的执行计划。

MPP 的主要问题是不能获得很好的可扩展性，而可扩展性是大数据处理的关键。

2. 内存计算

内存计算是指在内存数据库上进行数据存储和处理。内存和闪存技术快速发展，数据库厂商推出的新一代数据库技术是基于内存和闪存的数据一体机，如 SAP HANA、Oracle Exadata 和 IBM PureData，这些系统统称为内存计算系统。

（1）内存计算的特点

21 世纪第 1 个十年间，系统内存成本急剧下降，64 位的服务器能够访问数量非常大的系统内存，已经给现有的商业软硬件以新的发展机会。利用硬件价格的下降，可以利用高速的 CPU、计算机加载的大量内存作为一个数据运行的基础。内存计算允许在服务器的内存中处理大量的实时数据，提供即时分析和交易的结果。内存计算的优点如下。

- **加速数据访问**。内存计算在计算数据的过程中没有磁盘 I/O，内存计算比磁盘访问快 100 百万倍，传统数据库磁盘读取 5 毫秒，内存数据库磁盘读 5 纳秒。可以利用内存的高速性能，更快速地获取数据、汇总数据、分析数据。
- **实现大规模数据的分而治之**。在大数据时代，一台服务器计算能力已经不够用，但可以通过分而治之的方式，通过分布式的环境，内存计算的服务器被分到不同的节点上，快速地进行数据分散计算、数据分散汇总、获取更快速的结果。
- **良好的集成性**。内存计算具有原生支持行和列数据存储的功能，提供全面的 ACID（原子性、一致性、隔离性、持久性）会话功能。计算、规划引擎和数据存储库放在一起，提供业务信息的持久性视图以及统一的信息建模设计环境。数据管理服务包括 SQL 和其他接口，访问结构化和非结构化数据存储的数据集成能力。
- **数据压缩**。内存计算对数据压缩要求很高。更高压缩比的数据能够更多地进入内存进行计算。

（2）内存计算的工作原理

内存计算的工作原理如图 3-32 所示。

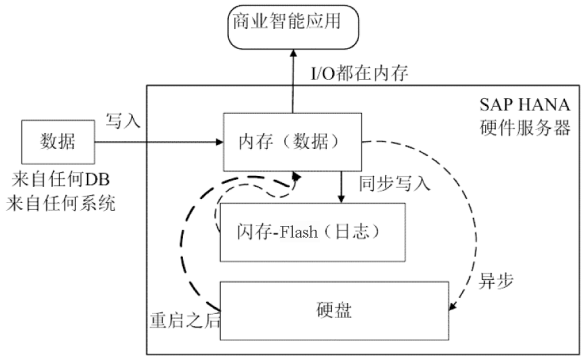


图 3-32　内存计算的数据写入（以 SAP HANA 为例）

数据读取。内存计算在计算（读取、聚合、计算结果等）数据的过程中没有磁盘 I/O。除非数据库表中的记录不在内存中（被手动卸载了）。

数据写入。不管使用什么接口，使用 JDBC/ODBC/ IMPORT 等命令，或者使用 ETL 工具，写入数据到内存计算平台时，数据会先写入内存，同时写入日志闪存。虽然写入内存是非常快的，但是闪存的写入速度才是决定内存计算系统对于数据真正的写入速度。这

就是为什么要用写入效率最好的 FUSION-IO SSD 和 PCI-E 接口的闪存原因。SSD 写入和读取的效率基本上取决于驱动程序写的好坏。

数据异步到磁盘。数据写入内存，日志同时写入闪存，然后内存中的数据间隔大约默认每 300 秒就会同步到磁盘。如果在 300 秒的这个异步复制到来之前掉电了，就要依靠日志进行恢复。这个异步存储主要是为了加速恢复的过程，让磁盘二进制文件直接导入内存，而最小化重现日志。备份文件和备份目录一般放在内存计算一体机的硬件服务器之外进行存储，以防止一体机服务器物理损毁。

目前，内存计算的解决方案往往是硬件和软件组合起来的一体机解决方案，使得客户分析海量数据，而且是以接近实时的方式来同步数据的，不需要花费太多时间在数据传输上。最大的内存计算平台已经能支持 100TB 的内存计算了。

3. 流计算

传统的数据操作，首先将数据采集并存储在数据库中，然后通过查询和数据库进行交互，得到用户想要的信息。整个过程中，用户是主动的，而数据库系统是被动的。但是，对于现在大量存在的实时数据，比如股票交易的数据等，数据实时性强、数据量大且不间断，传统的架构已经不合适。这种实时数据被称为流数据（Stream）。

流计算（Stream Computing）就是专门针对这种实时数据类型（即流数据）准备的。在流数据不断变化运动的过程中实时地进行分析，捕捉到可能对用户有用的信息，并把结果迅速发送出去。例如，为了支持个性化搜索广告，系统需要实时处理来自几百万唯一用户每秒成千上万次的查询，并即时分析用户的会话特征来提高广告相关性和预测模型的准确度。从 2010 年开始，流计算逐渐成为大数据处理中的应用热点。典型应用场合有证券数据分析、网站广告的上下文分析、社交网络的用户行为分析等。

流计算没有明确的概念，一个典型的流计算模型可以看作是一系列算子（点）和数据流（边）组成的数据流图。在该模型中，以数据流（边）的形式，多个处理单元（点）中的计算机程序对实时数据进行处理和传递。流数据的特点是数据实时处理，不同事件到达某一节点的顺序是不可控制的。

在通常的流计算系统中，数据以流的方式进入计算集群，集群中的处理单元对实时数据流进行提取、过滤和分析等操作，最后输出计算结果。一部分流计算框架包括控制节点，负责失效处理、负载均衡和节点管理等功能；而有的流计算框架采用无中心架构，各节点

地位平等，不能互相干涉。一个好的流计算框架的特点包括如下。

- 高性能，包括高吞吐量和低时延。
- 高可用性，完善的故障处理机制。
- 良好的伸缩性。
- 功能可扩展，易于二次开发。
- 提供友好的编程接口。
- 良好的负载均衡、Web 管理、自动部署等功能。

流计算框架和以 Hadoop MapReduce 为代表的批处理的比较如表 3-14 所示。

表 3-14　Hadoop 和流计算的比较

Hadoop	流计算
• 主要解决静态数据的批处理 • 注重总吞吐量 • 处理流数据时需要将数据流缓存起来并进行分块处理。但是，若分块太小，则系统开销明显增大；而分块太大，则无法满足实时性的要求	• 无须数据准备时间，对数据流进行实时处理 • 注重数据处理的低时延

主要的流计算框架有 Apache S4（开源，源于 Yahoo！的广告业务）、Storm（由 Twitter 开源）、HStreaming（Hadoop）、Esper/NEsper、Kafka（Linkedin）、Scribe（Apache，日志）、Flume（Apache，日志）、DStream（百度）、DSpark（豆瓣）、Infosphere Streams（IBM）、SteamBase 等。除此之外，学术界也诞生了一些流计算框架，如 Spark、Esc 等。

（1）S4

S4（Simple Scalable Streaming System）是 Apache 上由雅虎发布的开源流计算平台，它是一个分布式流处理引擎，开发者可以在这个引擎基础上开发面向无界的、不间断的流数据处理应用。S4 是通用、可扩展性良好、具有分区容错能力、支持插件的分布式流计算平台，开发语言为 Java。

S4 将一个流抽象为由(K,A)形式的元素组成的序列，这里 K 和 A 分别是关键字和属性。在这种抽象的基础上，S4 设计了能够消费和发出这些(K,A)元素的组件，也就是 Process Element（PE）。Process Element 在 S4 中是最小的数据处理单元，每个 PE 实例只消费事件

类型、属性 Key、属性 Value 都匹配的事件，并最终输出结果或者输出新的(K,A)元素。

S4 将流的处理分为多个流事件，每个流事件都用(K,A)的形式表示，抽象为处理图中的有向边；因为流被分为多个流事件，就需要对应多个处理单元，每个 PE 唯一处理一种事件，并且 PE 间独立，大大降低了概念复杂性和系统复杂性。开发者要做的就是定制个性化的 PE。多个 PE 间有事件传递的依赖性，就形成集群。在 S4 的论文[16]中，给出一个基于 S4 的单词记数的实现过程，如图 3-33 所示。

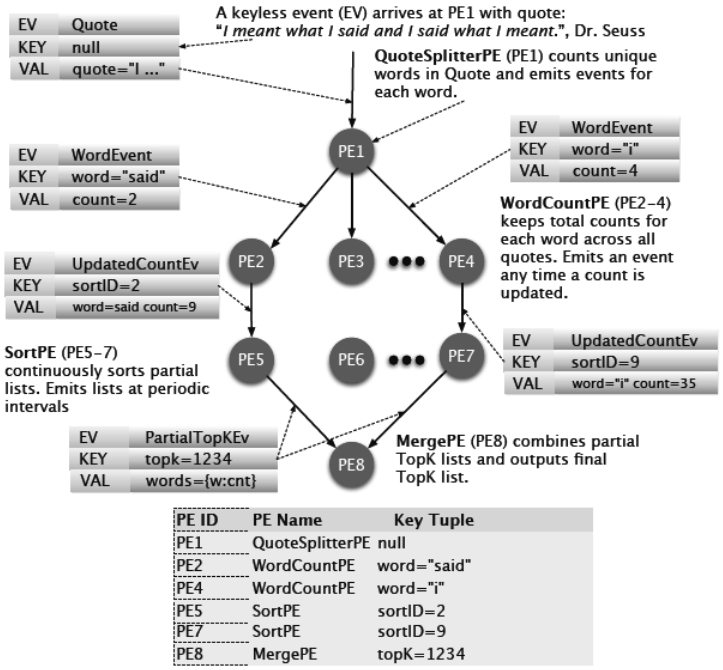


图 3-33　基于 S4 的单词记数示例

S4 使用 Processing Element Container（PEC）来表示集群，将多个 PE 包含到同一个 Container 中，PEC 接收源事件，并最终发送结果事件。S4 采用了对等架构，集群中的所有处理节点都是等同的，没有中心控制。这种架构将使集群的扩展性很好，处理节点的总数理论上无上限；同时，S4 将没有单点容错的问题。PN 负责监听事件，在到达事件上执行操作（PE 完成），然后通过通信层的协助分发事件，也可以发出输出事件。PEC 加上通

[16] http://www.4lunas.org/pub/2010-s4.pdf。

信处理模块就形成了 PE 的逻辑主机 Processing Node（PN）。PN 负责监听事件，在到达事件上执行操作（PE 完成），然后通过通信层的协助分发事件，也可以发出输出事件，如图 3-34 所示。

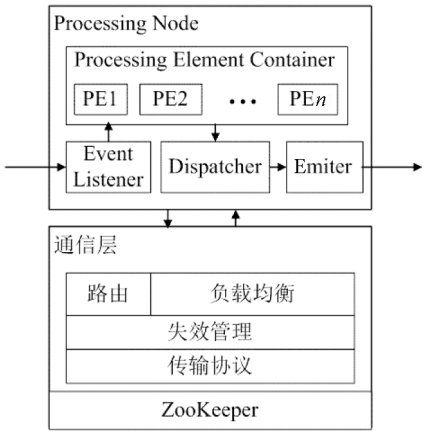


图 3-34　S4 的处理节点

S4 通过一个 hash 函数，将事件路由到目标 PN 上，这个 hash 函数作用于事件的所有已知属性值上（需要配置），所以一个事件可能被路由到多个 PN 上。然后 PN 中的事件监听器会将到来的事件传递给 PEC，PEC 以适当的顺序调用适当的 PE（每个被赋予关键字的 PE 都会被映射到一个确定的 PN 上，即图中的 PE 并不是物理存在一个 PN 相关，而是逻辑相关）。处理完成后，PN 可能发出输出事件，也可能向通信层请求协助向指定逻辑节点发送消息。

通信层提供了集群管理、故障恢复到备用节点、逻辑节点到物理节点映射的作用。同时通信层还使用一个插件式的架构来选择网络协议，使用 ZooKeeper 在 S4 集群节点之间协调。

（2）Storm

Storm 是 Twitter 的开源流计算平台。Storm 通过简单的 API 使开发者可以可靠地处理无界持续的流数据，进行实时计算，开发语言为 Clojure 和 Java，非 JVM 语言可以通过 stdin/stdout 以 JSON 格式协议与 Storm 进行通信。Storm 的应用场景很多：实时分析、在线机器学习、持续计算、分布式 RPC、ETL 处理等。

◎　Storm 的数据结构

流（Stream）。流（Stream）是 Storm 的核心概念，流是一个不间断的无界的连续元组

（Tuple）。

元组（Tuple）。Storm 在建模事件流时，把流中的事件抽象为元组（Tuple）。一个 Tuple 就是一个值列表，列表中的每个 Value 都有一个 Name，并且该 Value 可以是基本类型、字符类型、字节数组等，当然也可以是其他可序列化的类型。

源头（Spout）。Storm 认为每个 Stream 都有一个 Stream 源，也就是原始元组的源头，它将这个源头抽象为 Spout。Spout 可能是连接 Twitter API 并不断发出 Tweets，也可能是从某个队列中不断读取队列元素并装配为 Tuple 发射。

阀门（Bolt）。Storm 将流的中间状态转换抽象为 Bolt，Bolt 可以消费任意数量的输入流，只要将流方向导向该 Bolt，同时它也可以发送新的流给其他 Bolt 使用，这样一来，只要打开特定的 Spout（源头），再将 Spout 中流出的 Tuple 导向特定的 Bolt，Bolt 对导入的流做处理后再导向其他 Bolt 或者目的地。Bolts 可以将一个输入流转换为一个新的流。例如，可以转变一个消息到一个热门主题。Bolts 也可以做复杂的流转换，如从数据流中计算出流行趋势的流需要多个步骤，因此需要多种 Bolts。

Spout 和 Bolt 提供的接口，可以让应用程序继承这些接口并来具体实现，如图 3-35 所示。

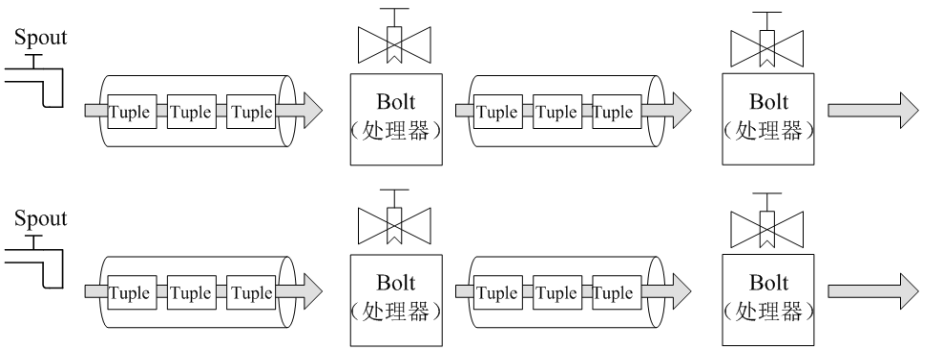


图 3-35　Storm 的基本数据结构

流拓扑（Topology）。多个步骤的流转换可以打包成拓扑 Topology，这是 Strom 集群执行的最高级的抽象。Topology 是一个流转换图，其中每个节点是一个 Spout 或 Bolt。在图 3-36 中的边表 Bolt 订阅那些流。当 Spout 或 Bolt 发出一个元组到流，它发送到每一个订阅该流的 Bolt。

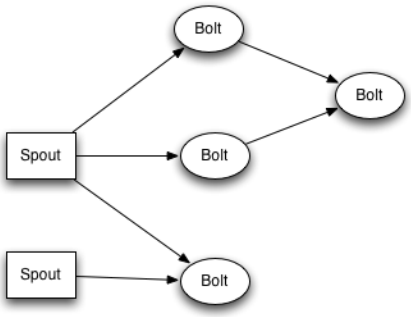


图 3-36　Storm 的拓扑

Storm 中拓扑定义仅仅是一些 Thrift 结构体，可以使用其他语言来创建和提交拓扑。

◎　Storm 集群

Storm 集群有两种节点：主节点（Master）和工作者节点（Worker）。主节点运行一个称之为 Nimbus 的后台程序，它类似于 Hadoop 的 JobTracker。Nimbus 负责在集群范围内分发代码、为 Worker 分配任务和监测故障。每个工作者节点运行一个称为 Supervisor 的后台程序。Supervisor 监听分配给它所在机器的工作，基于 Nimbus 分配给它的事情来决定启动或停止工作者进程。每个工作者进程执行一个 Topology 的子集（也就是一个子拓扑结构）；一个运行中的 Topology 由许多跨多个机器的工作者进程组成。

任何流在 Storm 中以分布式的方式并行运行，Topology 处理消息永远保持。Spout 和 Bolt 在集群中的许多线程中执行，它们在一个分布式系统中相互传递信息。信息从来没有通过任何中央路由器，而且没有中间队列。Storm 保证每个经过的 Topology 的消息都将被处理，Storm 实现无须任何中间排队，这是它处理速度快的关键。

Nimbus 和 Supervisor 之间所有的协调工作是通过 ZooKeeper 集群来进行的，如图 3-37 所示。Nimbus 守护进程和 Supervisor 守护进程是无法连接和无状态的；所有的状态维持在 ZooKeeper 中或保存在本地磁盘上。可以杀死 Nimbus 或 Supervisor 进程，它们不需要做备份。 这种设计导致 Storm 集群具有很好的稳定性。

Storm 擅长处理如下应用。

◎　流处理（Stream processing）。Storm 可用来实时处理新数据和更新数据库，兼具容错性和可扩展性。

- 连续计算（Continuous computation）。Storm 可进行连续查询并把结果即时反馈给客户端。比如把 Twitter 上的热门话题发送到浏览器中。
- 分布式远程程序调用（Distributed RPC）。Storm 可用来并行处理密集查询。Storm 的拓扑结构是一个等待调用信息的分布函数，当它收到一条调用信息后，会对查询进行计算，并返回查询结果。Distributed RPC 可以做并行搜索或者处理大集合的数据。

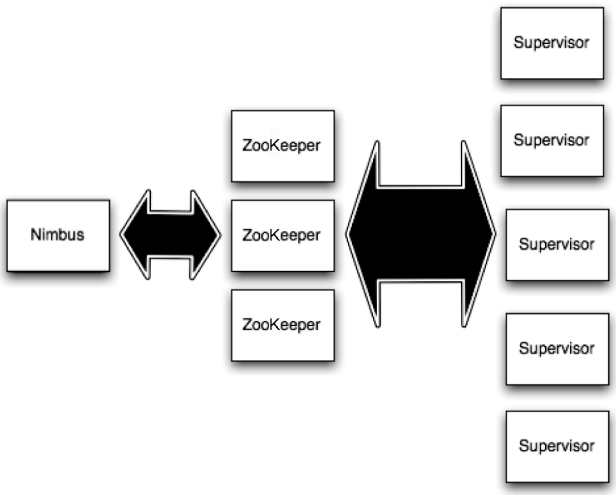


图 3-37　Storm 集群

3.3　大数据查询和分析技术：SQL on Hadoop

由于目前的大数据存储都不是基于关系型数据库的，所以传统的通过 SQL 语言来操作数据的方式无法直接使用，例如，对于 Hadoop 存储的数据是无法直接通过 SQL 来查询的。企业由于已经适应了在小数据上的灵活处理方式，转到 Hadoop 一下子变得无所适从。

传统数据库和数据仓库厂商（如 Oracle、Teradata 和 MySQL 等）也在研究解决办法，它们的思路是 Hadoop 将 MapReduce 的结果存储到 RDBMS 中，然后查询 RDBMS，但这样的解决方案牺牲了 Hadoop 的高效性。

为了让 SQL 专业分析人员能够通过 SQL 语言来操作和分析大数据，SQL on Hadoop

技术发展了起来。SQL on Hadoop 是直接建立 Hadoop 上的 SQL 查询，既保证 Hadoop 性能，又利用 SQL 的灵活性。SQL on Hadoop 正处于起步阶段，Hadoop 解决方案对于 SQL 语言支持的深度与广度各不相同，技术实践方式也很多样。最基本的工作是把传统的 SQL 语言进行中间转换后进行操作，例如 Hadoop 中的 Hive，就是把 SQL（HiveQL，Hive 的类 SQL 语句）编译成 MapReduce，从而读取和操作 Hadoop 上的数据。这是很多 SQL on Hadoop 技术的基础，它提供了一种能力，让企业把信息管理能力从结构化数据延伸到非结构化数据。SQL on Hadoop 技术还在不断发展，IT 厂商也推出了很多对 Hive 的扩展，大致分为 4 种情况。

一是基于 PostgreSQL 的 Hadoop 分析，如 Hadapt 的技术。该方法也称为 DB on TOP，即组合利用不同的计算框架面向不同的数据操作，同时解决结构化与非结构化数据。最早由 Hadapt 公司在 2010 年提出，其中以 EMC Greenplum HAWQ、Hadapt、Citus Data 为代表。Hadapt 以 PostgreSQL 架构在 Hadoop 上，来完成对结构化数据的查询。它提供了统一的数据处理环境，利用 Hadoop 的高扩展性和关系数据库的高速性，分开执行 Hadoop 和关系型数据库之间的查询。Citus Data 则通过把多种数据类型转化成数据库的原生类型，运用分布式处理技术来完成查询。

二是 Hive 的性能改进和优化，如 Stinger/Tez，Cascading Lingual 也类似 Hive，将 ANSI SQL 编译成 MapReduce。Hortonworks 的 Stinger 通过对原生态 Hive 做改造，优化 SQL 查询速度，使其达到 5～30 秒，完成对 SQL 查询。

三是为 HBase 建立 SQL 层，如 DRAWN Scale 和 Saleforce Phoenix。

四是实时互动 SQL 分析，如 Apache 的 Drill 和 Impala。Apache 的 Drill 项目，因开放的数据格式和查询语言，已经获得了专业的 Hadoop 商业发行版供应商 MapR 的支持。

3.3.1　Hive：基本的 Hadoop 查询和分析

Hive 是由 Facebook 开发的，是用来管理结构化数据的中间件，是 Hadoop 上的数据仓库基础构架。它架构在 Hadoop 之上，以 MapReduce 为执行环境，数据储存于 HDFS 上，元数据储存于 RDMBS 中。它提供了一系列的工具，可以用来进行数据提取转化加载（ETL），这是一种可以存储、查询和分析储存在 Hadoop 中的大规模数据的机制。

Hive 以人们熟悉的 SQL 作为数据仓库的工具，并具有很好的可扩展性和互操作性，可扩展性表现在它能采用自选语言所开发的可插入式的 MapReduce 脚本及丰富的用户定

义的数据类型和用户定义的函数，互操作性表现为它是一个可扩展的框架以支持不同的文件和数据格式。

1. Hive 的体系架构

Hive 主要分为以下 3 个部分，如图 3-38 所示。

Hive 体系架构

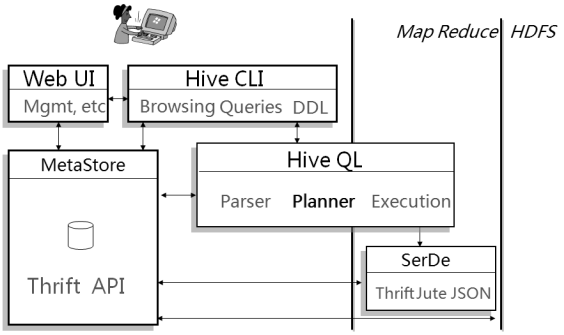


图 3-38　Hive 体系架构

（1）用户接口

用户接口主要有 3 个：Hive 命令行接口 CLI、Client 和 Web UI。

命令行接口（CLI）

最常用的是 CLI，CLI 启动的时候会同时启动一个 Hive 副本。CLI 主要包括如下内容。

- DDL（数据描述语言）
  - 生成表：create table，放弃表：drop table，表改名：rename table。
  - 变更表：alter table，增加列：add column。
- Browsing（浏览）
  - show tables
  - describe table
  - cat table

- 查询 Queries
- 装载数据 Loading Data

Client

Client 是 Hive 的客户端，用户通过 Client 连接至 Hive Server。在启动 Client 模式的时候，需要指出 Hive Server 所在的节点，并且在该节点启动 Hive Server。Hive 支持多种数据库的整合，可通过 JDBC/ODBC 等驱动器来访问 Hive，包括：

- Hive JDBC Driver (Java)
- Hive ODBC Driver (C++)
- Hive Add-in for Excel (by Microsoft)
- Thrift（C/C++、Python、Perl、PHP 等）

Web UI

Web UI 是通过浏览器访问 Hive。主要有：

- MetaStore UI，它能浏览和导航系统中所有的表，给每个表和每个列做注释，也能抓取数据的依赖关系。
- HiPal，它能通过鼠标点击交互式地构建 SQL 查询，并支持投影、过滤、分组和合并。

（2）元数据存储（MetaStore）

- 存储表/分区的属性

Hive 将元数据存储在关系型数据库，如 MySQL、Derby 等，或者在一个文本文件中。Hive 中的元数据包括表的 Schema、序列化和反序列化 SerDe 库，表的名字和属性（是否为外部表等），表的列和分区及其属性，表的数据所在 HDFS 的目录等信息。

- Thrift API

当前客户机的 PHP（Web 接口）、Python（旧的 CLI）、Java（查询引擎和 CLI）、Perl（Tests）等 Thrift API。

（3）完成 Hive QL 查询的解释器、编译器、优化器、执行器

解释器（Parser）、编译器（Compiler）、优化器（Optimizer）完成 Hive QL 查询语句从词法分析、语法分析、编译、优化以及查询计划的生成。生成的查询计划存储在 HDFS 中，并在随后有 MapReduce 调用执行器（Executor）执行。

Hive 的数据存储在 HDFS 中，大部分的查询由 MapReduce 完成（不包含 * 的查询，比如 select * from tbl 不会生成 MapRedcue 任务），如图 3-39 所示。

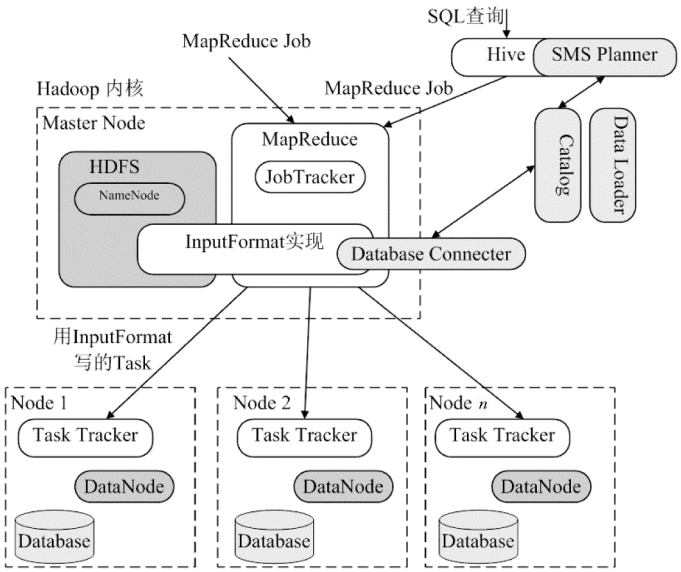


图 3-39　Hive 与 Hadoop 平台的关系

2. Hive 的数据类型和数据模型

Hive 支持的 Types 类型包括简单类型和复杂类型。简单类型包括整数（Tinyint、Smallint、Int 和 Bigint）、布尔型（Boolean）、浮点数（Float、Double）和字符串（String）。复杂类型包括 Structs: {a INT; b INT}、Maps: M['group']、Arrays: ['a', 'b', 'c']。

Hive 中所有的数据都存储在 HDFS 中，Hive 中包含以下数据模型：Table、External Table、Partition、Bucket。

（1）表（Table）

Hive 中的 Table 和关系型数据库中的 Table，在概念上是类似的，每一个 Table 在 Hive 中都有一个相应的 HDFS 中的目录存储数据。例如，一个表 pvs，它在 HDFS 中的路径为：

/wh/pvs，其中，wh 是在 hive-site.xml 中由 ${hive.metastore.warehouse.dir} 指定的数据仓库的目录，所有的 Table 数据（不包括 External Table）都保存在这个目录中。

例如，生成一个名为 t1 的表，它的 ds 列是字符串型，ctry 列是浮点型，li 列是一个复杂数据类型，Hive QL 语句如下：

CREATE TABLE t1(ds string, ctry float, li list<map<string,struct<p1:int, p2:int>>);

（2）分区（Partition）

Partition 类似于关系型数据库中的 Partition 列的密集索引，但是 Hive 中 Partition 的组织方式和数据库中的有很大不同。在 Hive 中，表中的一个 Partition 对应于表下的一个目录，所有的 Partition 的数据都存储在对应的目录中。例如：pvs 表中包含 ds 和 ctry 两个 Partition，则对应于 ds = 20090801，ctry = US 的 HDFS 子目录为：/wh/pvs/ds=20090801/ctry=US；对应于 ds = 20090801，ctry = CA 的 HDFS 子目录为：/wh/pvs/ds=20090801/ctry=CA。

例如，生成一个表 t1，ds 列的类型是字符串，hr 列的类型是整数，并对 ds 和 hr 列进行分区和相应的操作，Hive QL 语句如下：

```
CREATE TABLE t1(ds string, hr int)
PARTITIONED BY (ds string, hr int);
INSERT OVERWRITE TABLE
t1 PARTITION(ds='2009-01-01', hr=12)
SELECT * FROM t1;
ALTER TABLE t1
ADD PARTITION(ds='2009-02-02', hr=11);
SELECT * FROM t1 WHERE ds='2009-02-02' AND hr=11;
```

（3）存储桶（Bucket）

Bucket 对指定列计算哈希值，根据哈希值切分数据，目的是为了并行，每一个 Bucket 对应一个文件。例如，将 user 列分散至 32 个 Bucket，首先对 user 列的值计算哈希值，对应哈希值为 0 的 HDFS 目录为：/wh/pvs/ds=20090801/ctry=US/part-00000；哈希值为 20 的 HDFS 目录为：/wh/pvs/ds=20090801/ctry=US/part-00020。

（4）外部表（External Table）

External Table 指向已经在 HDFS 中存在的数据目录，它可以创建 Table 和 Partition，表中数据假设是用 Hive 兼容的格式。它和 Table 在元数据的组织上是相同的，而实际数据的存储则有较大的差异。Table 的创建过程和数据加载过程是两个过程（但这两个过程

可以在同一个语句中完成）。在加载数据的过程中，实际数据会被移动到数据仓库目录中；之后对数据的访问将会直接在数据仓库目录中完成。删除表时，表中的数据和元数据将会被同时删除。但是，External Table 只有一个过程，加载数据和创建表同时完成（CREATE EXTERNAL TABLE ……LOCATION），实际数据是存储在 LOCATION 后面指定的 HDFS 路径中，并不会移动到数据仓库目录中。当删除一个 External Table 时，仅删除元数据，表中的数据不会真正被删除。

例如，创建一个 External Table，名称为 et1，列 c1 类型是字符串，列 c2 类型是整数，External Table 存储位置是/user/mytables/mydata，语句如下：

```
CREATE EXTERNAL TABLE et1(c1 string, c2 int)
LOCATION '/user/mytables/mydata';
```

3. Hive QL

Hive 定义了简单的类 SQL 查询语言，称为 Hive QL，也缩写为 HQL。它既允许熟悉 SQL 的用户查询数据，同时，这个语言也允许熟悉 MapReduce 的开发者开发自定义的 Mapper 和 Reducer 来处理 MapReduce 的 Mapper 和 Reducer 自身无法完成的复杂的分析工作。

Hive QL 的常用查询操作主要有：ANSI JOIN（只有 equi-join）、多个表 Insert、多个表的 Group by、Sampling 等。例如，Join 和 Group by 如下。

（1）Join 操作

Join 操作如图 3-40 所示。

page_view

pageid	**userid**	time
1	**111**	9:08:01
2	**111**	9:08:13
1	**222**	9:08:14

X

user

userid	age	gender
111	25	female
222	32	male

=

pv_users

pageid	age
1	25
2	25
1	32

Hive QL:

```
INSERT INTO TABLE pv users
SELECT pv.pageid, u.age
FROM page_view pv JOIN user u ON (pv.userid = u.userid);
```

图 3-40　Join 操作

例如，上面的语句表示一个 3 列的表 page_view（访问页 id、用户 ID 和访问时间）和 3 列的表 user（用户 ID、年龄和性别），通过相同的用户 ID 执行 Join 的操作，形成一个新的两列的表 pv_user，展示出访问页面的用户的年龄结构。

（2）Group by 操作

Group by 操作如图 3-41 所示。

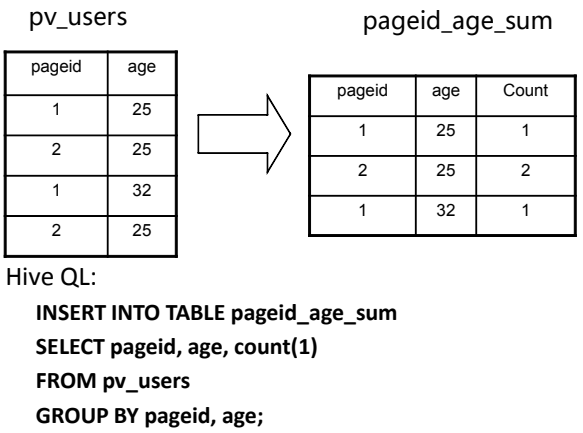


图 3-41　Group by 操作

4. Hive 的用户扩展性

Hive 是一个很开放的系统，很多内容都支持用户定制。Hive 扩充性方面不仅表现为对多种类型的支持，也表现在文件格式、脚本、函数等的自定义支持上，因此它能实现性能与水平扩展能力兼具。

（1）文件格式

Hive 没有专门的数据存储格式，也没有为数据建立索引，用户可以非常自由地组织 Hive 中的表，只需要在创建表的时候告诉 Hive 数据中的列分隔符和行分隔符，Hive 就可以解析数据。Hive 可以很好地工作在 Thrift 之上，也允许用户指定数据格式。用户定义数据格式需要指定 3 个属性：列分隔符（通常为空格、“\\t”、“\\x001”）、行分隔符（“\\n”）以及读取文件数据的方法（Hive 中默认有 3 个文件格式 TextFile、SequenceFile 以及 RCFile，如表 3-15 所示）。

表 3-15　Hive 支持的文件格式

	TextFile	SequenceFIle	RCFFile
数据类型	只是文本	文本/二进制	文本/二进制
内部的存储顺序	基于行	基于行	基于列
压缩	基于文件	基于块	基于块
可分裂	是	是	是
压缩后可分裂	不可以	可以	可以

下面的命令创建一个表，以 TextFile 进行存储：

```
CREATE TABLE mylog ( user_id BIGINT, page_url STRING, unix_time INT)
STORED AS TEXTFILE;
```

当用户的数据文件格式不能被当前 Hive 所识别的时候，可以自定义文件格式。可以参考 contrib/src/java/org/apache/hadoop/hive/contrib/fileformat/base64 中提供的例子。写完自定义的格式后，在创建表的时候指定相应的文件格式就可以了：

```
CREATE TABLE base64_test(col1 STRING, col2 STRING)
STORED AS
INPUTFORMAT 'org.apache.hadoop.hive.contrib.
fileformat.base64.Base64TextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.contrib.
fileformat.base64.Base64TextOutputFormat';
```

（2）序列化

当进程在进行远程通信时，彼此可以发送各种类型的数据，无论是什么类型的数据都会以二进制序列的形式在网络上传送。发送方需要把对象转化为字节序列才可在网络上传输，称为对象序列化；接收方则需要把字节序列恢复为对象，称为对象的反序列化。SerDe 是 Serialize/Deserilize 的简称。Hive 具有通用的序列化和反序列化接口 SerDe，是非结构化数据和结构化数据间转换的灵活的接口。用户在建表时可以使用 Hive 自带的 SerDe 或用自定义的 SerDe。目前存在的一些 SerDe 如表 3-16 所示。

Hive 序列化的格式包括：

- 分隔符（Tab、逗号、CTRL-A）。
- Thrift 协议。

Hive 的反序列化是对<key,value>反序列化成 Hive Table 的每个列的值。反序列化格式（内存内）包括：

- Java Integer/String/ArrayList/HashMap。
- Hadoop Writable 类。
- 用户自定义类。

表 3-16　目前存在的一些 SerDe

	LazySimpleSerde	LazyBinarySerde （HIVE–640）	BinarySortableSerde
序列化格式	分隔符	专有的 Binary	专有的 Binary Sortable*
反序列化格式	LazyObject*	LazyBinaryObject*	Writable
	ThriftSerde (HIVE-706)	RegexSerde	ColumnarSerde
序列化格式	依靠 Thrift 协议	Regex formatted	基于专有列
反序列化格式	用户定义的类， Java Primitive Objects	ArrayList<String>	LazyObject*

SerDe 位于'hive_contrib.jar 文件中。

（3）MapReduce 脚本的使用

下面是一个 HQL 使用 MapReduce 脚本的例子：

```
FROM (FROM pv_users
    MAP pv_users.userid, pv_users.date USING 'map_script' AS (dt, uid)
    CLUSTER BY dt) map
INSERT INTO TABLE pv_users_reduced
REDUCE map.dt, map.uid USING 'reduce_script' AS (date, count);
```

为了执行定制的 Mapper 和 Reducer 脚本，用户可以提交下列命令：

```
SELECT TRANSFORM(user_id, page_url, unix_time) USING  'page_url_to_id.py'
AS (user_id, page_id, unix_time)
```

它使用 TRANSFORM 子句来嵌入命令行 Mapper 和 Reducer 脚本。

（4）用户自定义函数的使用

Hive 内置的函数如下。

- 数学：round、floor、ceil、rand、exp……
- 集合：size、map_keys、map_values、array_contains……
- 类型转换：cast。
- 日期：from_unixtime、to_date、year、datediff……
- 条件：if、case、coalesce……
- 字符串：length、reverse、upper、trim……

用户可以自定义函数，例如，用户自定义如下一个 Java 类：

```
package com.example.hive.udf;
import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;
public final class Lower extends UDF {
  public Text evaluate(final Text s) {
    if (s == null) { return null; }
    return new Text(s.toString().toLowerCase());
  }
}
```

登记这个类，采用如下的命令：

```
CREATE FUNCTION my_lower AS 'com.example.hive.udf.Lower
```

用 SQL 语句使用这个类：

```
SELECT my_lower(title), sum(freq) FROM titles GROUP BY my_lower(title);
```

Apache[17]提供了 Hive 的源代码下载、入门指南[18]、教程[19]和语言指南[20]供学习者下载，读者可以详细学习和参考。

[17] http://hive.apache.org/。

[18] https://cwiki.apache.org/confluence/display/Hive/GettingStarted。

[19] https://cwiki.apache.org/confluence/display/Hive/Tutorial#Tutorial-Custommap%2Freducescripts。

[20] http://wiki.apache.org/hadoop/Hive/LanguageManual。

3.3.2　Hive 2.0：Hive 的优化和升级

自从 2007 年 Facebook 提出 Apache Hive 和 HiveQL 后，它们已经成为事实上的 Hadoop 上的 SQL 接口。如今，各种类型的大公司或小公司都在使用 Hive 这种非常普遍的方法来访问 Hadoop 数据，从而给公司或者用户带来更多的价值。同时，还有许多公司通过大量已存的 BI 工具生态系统来达到相同的目的，这些 BI 工具同样使用 Hive 作为接口。Hive 用于建立大规模的批量计算，这在数据报告、数据挖掘以及数据准备等应用场景很有效。这些应用场景很重要，但是 Hadoop 的需求十分广阔，企业用户越来越需要 Hadooop 具备更高的实时性和交互性[21]。

由于 Hive 对 MapReduce 依赖，查询速度有着“先天性不足”，因为在查询的过程中，MapReduce 需要扫描整个数据集，而且在作业的处理过程中还需要把大量的数据传输到网络。浏览一个完整的数据集可能要花费几分钟到几小时，这完全是不切实际的。对主流用户而言，难以有很大的吸引力。

对 Hive 的优化和升级是一个重要的工作，主要包括 Stinger Initiative 和 Presto 等。

1. Stinger[22]

Stinger 是 Hortonworks 公司的 Apache 开源项目，是对 Hive 进行优化的项目。它主要的改进如下。

（1）库优化：智能优化器

- 生成简化的有向无环图（DAG）。
- 引入 in-memory-hash-join。适用于有一方适合在内存中的 join，这是一个全新的 in-memory-hash-join 算法，借此算法 Hive 可以把小表读到哈希表中，可以遍历大文件来产生输出。
- 引入 Sort-Merge-Bucket Join。适用于表在同样的关键词上被分为 bucket 的情形，在速度改进方面是巨大的。
- 减少在内存中的事实表的足迹。

[21] http://hortonworks.com/blog/100x-faster-hive/。

[22] http://www.adattarhazforum.hu/letoltes/2013_dw_forum/adattarhaz_forum_2013_hortonworks_chris_harris_stinger_initiative.pdf。

- 让优化器自动挑选 map joins。

（2）多维度的结构化数据

在 Hive 中采用企业级数据仓库（EDW）中很普通的维度模式，产生大的数据表和小的维度表。维度表经常小到能适合 RAM。有时被称为 Star Schema。例如，在维度表上的查询（源自 TPC-DS Query 27）：

```
SELECT col5,avg(col6)
FROM fact_table
    join dim1 on(fact_table.col1=dim1.col1)
    join dim2 on(fact_table.col2=dim2.col1)
    join dim3 on (fact_table.col3=dim3.col1)
    join dim4 on (fact_table.col4=dim4.col1)
GROUP BY col5 ORDER BY col5 LIMIT 100;
```

（3）优化的列存储（ORCFile）

优化的列存储（ORCFile）包括如下内容。

- 生成一个更好的列存储文件，与 Hive 数据模型紧密一致。
- 把复杂的行类型分解为原始类型，便于更好地压缩和投影。
- 对必需的列，从 HDFS 中只读 bytes。
- 既储存文件也储存文件的每个节。
- 增加聚合函数，如 min、max、sum、average、count 等
- 允许通过排序列快速访问，能够快速校验是否一个值是存在的。

（4）深度分析能力

支持 SQL:2003 Window Functions。

- OVER 子句
  - 支持 Multiple PARTITION BY 和 ORDER BY。
  - 支持 Windowing (ROWS PRECEDING/FOLLOWING)。
  - 支持大量的聚合，RANK、FIRST_VALUE、LAST_VALUE、LEAD / LAG、Distrubutions 等。

（5）与 Hive 数据类型的一致性

数据类型的优化包括如下内容。

- 增加了固点 NUMERIC 和 DECIMAL 类型。
- 增加了有限域大小的 VARCHAR 和 CHAR 类型。
- 增加了 DATETIME。
- 对 FLOAT 增加大小到从 1～53。
- 增加了考虑兼容性的同义字，对应 BINARY 的 BLOB，对应 STRING 的 TEXT，对应 FLOAT 的 REAL。
- 增加了 SQL 语义，如更多地用 IN、NOT IN、HAVING 的子查询，EXISTS 和 NOT EXISTS 等。

（6）Stinger、Tez 和 YARN 的整合：Hadoop 2.0

Stinger 与 YARN 实现了有效集成，以实现真正的 Hadoop 2.0。Stinger 的优化工作仍在继续，第 2 阶段的工作包括：ORCFile 的继续完善；Hive 查询服务器的优化，如预热的 Container 和低时延的分发；对 Tez 的优化，如表达数据处理、任务更简化和消除磁盘写等。进一步，则会按照现代化处理器的体系结构进行优化，如实现向量查询引擎，以及为向量引擎所做的对缓存中访问数据的优化，如图 3-42 所示。

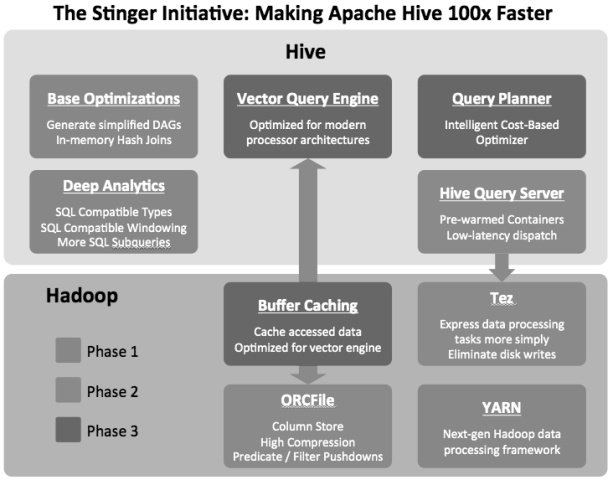


图 3-42　Stinger 项目与 Hadoop 2.0 的关系

Tez 是构架在 YARN 上的底层数据处理执行引擎[23]。它以 MapReduce、Hive、Pig、Cascading 为基础，实现作业的流水线操作，它删除了任务和作业的启动时间；在流水线的步骤之间，Hive 和 Pig 工具不再需要移动到队列的末尾；不再写中间结果到 HDFS 中，产生更少的硬盘和网络的使用。YARN 的 ApplicationMaster 运行一个 Tez 任务的有向无环图（DAG）。每个 Tez 任务是一个可插入的 Input、Processor 和 Output ，表示为<Input, Processor, Output>。Tez 任务可以是一个 MapReduce 的‘Map’或者一个 MapReduce 的‘Reduce’，也可以是中间的‘Reduce’，即 Map-Reduce-Reduce。对于 Pig/Hive，Tez 任务可以是特殊的 Pig/Hive ‘Map’、在内存中的 Map 或者特殊的 Pig/Hive ‘Reduce’等。

2. Presto

Hive 是 Facebook 专为 Hadoop 打造的一款数据仓库工具。因为它主要依赖 MapReduce 运行，随着大数据的发展，其在速度上已不能满足日益增长的数据要求。Facebook 最新的交互式大数据查询系统 Presto，类似于 Cloudera 的 Impala 和 Hortonworks 的 Stinger，解决了 Facebook 迅速膨胀的海量数据仓库快速查询需求。据 Facebook 称，使用 Presto 进行简单的查询只需要几百毫秒，即使是非常复杂的查询，也只需数分钟便可完成，它在内存中运行，并且不会向磁盘写入。

3.3.3 实时互动的 SQL：Impala 和 drill

1. Impala

Impala 是 Cloudera 公司的实时查询开源项目，据其官网的产品实测表明 Impala 比原来基于 MapReduce 的 Hive QL 查询速度提升 3～90 倍。Impala 参考了 Google Dremel，但在 SQL 功能上更胜一筹。Impala 是一种交互式的 SQL，它是为 Hadoop 上做低时延的 SQL 查询而专门设计的。它也采用接近 ANSI-92 标准的 Hive QL 来执行查询，具有对现有的 Hadoop 应用兼容的 SQL 接口。但是，Impala 不再使用较缓慢的 Hive+MapReduce 的批处理，而是使用与商用并行关系数据库中类似的分布式查询引擎。

Impala 架构如图 3-43 所示。Impala 采用的是通用的 HiveSQL 和 JDBC/ODBC 等接口，具有全局统一的元数据存储和调度，查询则是完全的分布式并行处理，对 HDFS 或者 HBase

[23] http://incubator.apache.org/projects/tez.html。

在本地直接读。Impala 分布式引擎由 Query Planner、Query Coordinator 和 Query Executor 3 部分组成，可以直接从 HDFS 或者 HBase 中用 Select、Join 和统计函数查询数据，从而大大降低了延迟。由于直接在 HDFS 或者 HBase 存储和元数据上做查询，它具有 Hadoop 的灵活性、横向扩充性和低成本优势，但不需要数据和元数据的复制/同步，在本地处理也有效避免了网络瓶颈。

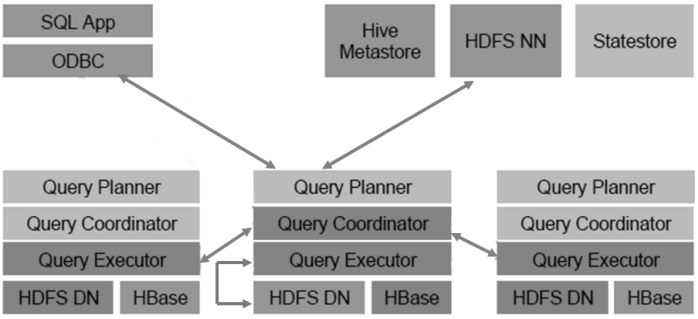


图 3-43　Impala 架构

因此，Impala 有两个基本单元：Statestore 和 Impalad。Impalad 是每个节点上处理数据的基本单元，它处理外部客户端的请求和与查询执行相关的内部请求，包括 Query Planner、Query Coordinator 和 Query Executor。客户端的 SQL 通过 ODBC/JDBC/HueGUI/Shell 等到达，Query Planner 把请求转换成计划片段的集合，Query Coordinator 则对分布的 Impalad 的本地数据启动执行。在执行期间，中间结果在各个 Query Executors 之间流动，查询结果最终被流动回客户端。

谷歌的 Dremel 之所以能在大数据上实现交互性的响应速度，是因为使用了两方面的技术：一是对有嵌套结构的嵌套关系型数据采用了全新的列式存储格式，另一个是分布式可扩展统计算法能够在几千台机器上并行计算查询结果。与 2010 年 Dremel 论文只能处理单表查询相比，Impala 已经能够支持完整的 Join 操作。此外，除了 Trevni 列式存储格式之外，Impala 还支持广泛的其他格式。Impala 并不会取代传统的数据仓库和 MapReduce+Hive。数据仓库在对数量有限的结构化数据集做复杂的分析处理时仍然更加适用，而持续运行的数据转换负载还是 MapReduce 的用武之地。

Impala 采用与 Hive 相同的元数据、SQL 语法、ODBC 驱动程序和用户接口（Hue Beeswax），这样在使用 Cloudera Hadoop（CDH）产品时，批处理和实时查询的平台是统一的。目前支持的文件格式是文本文件和 SequenceFiles（可以压缩为 Snappy、GZIP 和 BZIP，

前者性能最好）。其他格式如 Avro、RCFile、LZO 文本和 Doug Cutting 的 Trevni 将在正式版中支持。

2. Drill

大数据面临的一个很大的问题是大多数查询都是很缓慢而且非交互式的。谷歌的 Dremel[24]是为了解决这一问题研发的，它能以极快的速度处理网络规模的海量数据。据谷歌的研究报告显示，Dremel 能以 PB 数量级进行查询，而且查询只需几秒钟时间就能完成。而其对应的 Apache 开源版本就是 Drill。因此，Drill 是一个专门用于互动分析大型数据集的分布式系统。Drill 用标准的 SQL 来进行大数据的互动分析，客户端通过任何 SQL 工具输入一个 SQL 语句，通过 Drill 的 ODBC 驱动器连接到 Drill 的驱动，并经过 SQL 查询的 Parser 进行解析，经 Query Planner 进行计划，在节点的 Drillbit 上进行执行，多个 Drillbit 共同完成大规模的查询执行。

（1）体系架构

Drillbits 运行在每个节点上，这样的设计有助于最大化数据的本地性，实现本地数据本地处理。数据处理在 MapReduce（YARN 也能被支持）之外进行，查询能被输入到任何一个 Drillbit，协调、查询计划、优化、调度、执行都是分布的。默认情况下，Drillbit 拥有所有角色。发起的 Drillbit 担当领头的角色，管理查询的执行、调度本地化的信息等职责。任何节点都可以担当查询的结束点。ZooKeeper 负责协调集群成员之间的信息。分布式的缓存（Hazelcast）用于元数据、本地化信息等，Drillbit 的模块如图 3-44 所示。

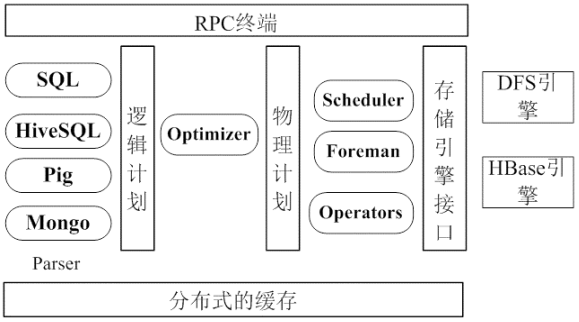


图 3-44　Drillbit 的模块

[24] http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/zh-CN//pubs/archive/36632.pdf。

Drill 体系架构包含 4 个关键部件或称为层。

◎　查询语言。

这一层，是为了解析用户查询并构建一个执行计划。初始的设计目标是支持被 Dremel 和 Google BigQuery 所适用的 SQL 类的查询语言，我们称为 DrQL。但 Drill 也被设计能支持其他语言和编程模型，如 Mongo Query Language、Cascading 或者 Plume。

如下面的 SQL 语句是一个典型的 Drill 查询：

```
SELECT * FROM oracle.transactions, mongo.users, hdfs.events LIMIT1
```

◎　低时延的分布式执行引擎。

这一层负责执行查询计划器所形成的物理计划。它提供可扩展性和在 10000 台服务器上做 PB 级数据查询所需要的有效的容错性。这个引擎是基于在分布式执行引擎（如 Dremel、Dryad、Hyracks、CIEL、Stratosphere 等）的研究的基础上形成的，它是列存储的，它能被附加的运算器和连接器进行扩展。基本的组成包括运算器、函数、写、扫描器等，如图 3-45 所示。

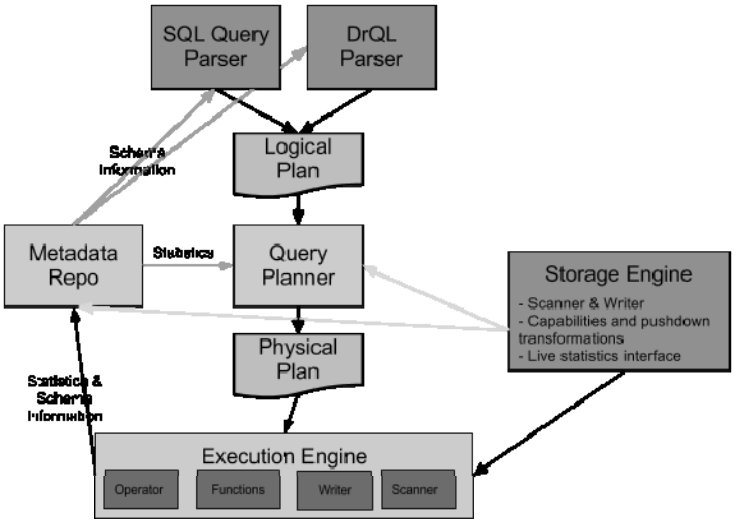


图 3-45　Drill 架构

◎　嵌套的数据格式。

这一层负责支持各种数据格式。初始设计目标是支持基于列的格式（用于 Dremel），

Drill 也被设计用于支持基于 schema 的格式，例如 Protocol Buffers/Dreme、Avro/AVRO-806/Trevni 和 CSV，以及 schema-less 格式，例如 JSON、BSON 或者 YAML。而且，它被设计为支持基于列的格式（Dremel、AVRO-806/Trevni 和 RCFile）和基于行的格式（Protocol Buffers、Avro、JSON、BSON 和 CSV）。一个特别突出的特点是 Drill 的执行引擎能足够灵活地既支持基于列的处理也支持基于行的处理。这是很重要的，因为在数据基于列的格式存储时基于列的处理能更有效，但是许多大数据资产是存储在基于行的格式上，在使用前需要进行转换。

- 可扩展的数据源。

这一层负责支持各种数据源，初始关注是利用 Hadoop 作为一个数据源。

（2）特点

Drill 具有查询快、体系结构开放、现代化等特点。

- 查询快。

用 Drill 做互动查询、数据分析、报表只需要 100 毫秒到 20 分钟时间。而如果用 MapReduce、Pig 和 Hive 来做数据挖掘、建模或大的 ETL，则需要 20 分钟到 20 小时。为什么能做到如此之快？一方面是因为 Drill 是分布式结构，但更重要的是 Drill 借鉴了谷歌的 Dremel/Bi BigQuery，具有延迟低、按照列来执行等特点，同时补充了与 MapReduce、Hive 和 Pig 的原始接口。

- 开放性。

体现在 Drill 是开源社区驱动的开源项目，在 Apache 软件基金会的支持下工作。

- 现代化。

Drill 采用标准的 ANSI SQL:2003，能提供相关的子查询和分析函数，弥补 SQL 类语句的不足。而且，Drill 能够使用任何基于 SQL 的工具，如 Tableau、Microstrategy、Excel、SAP 水晶报表、Toad、SQuirreL 等，具有标准的 ODBC 和 JDBC 的驱动。

Drill 支持嵌套和层次结构数据查询。嵌套数据正在变成流行：

- JSON、BSON、XML、Protocol Buffers、Avro……

例如：

JSON：

```
{
 "name": "Glenn",
 "gender": "Male",
 "followers":"100",
Children:{
            "name":"Sara",
            "name":"Jack"
            }
}
```

Avro：

```
enum Gender{
            MALE, FEMALE
             }
Record User{
        string name;
        Gender gender;
        long followers;
           }
```

- 数据源有意无意地支持嵌套。
  - MongoDB 天然支持嵌套数据。
  - 单独的 HBase 值可能是一个 JSON 文档（复合的嵌套类型）。
  - 谷歌 Dremel 的创新就在于高效的列存储和嵌套数据的查询。
  - 压平嵌套数据经常是容易出错的，也经常是不可能的，可能需要在每个层面思考重复和可选的域。
  - Apache Drill 一开始就支持嵌套数据（扩展到了 ANSI SQL:2003）。
- Drill 对 Schema 是可选的，支持无 Schema 和少 Schema 数据的查询。
  - 当前，许多数据源没有很严格的 Schema，Drill 能够支持这些数据查询。支持有 Schema 且快速变化的查询，支持每个记录有不同的 Schema 的查询，支持在 HBase、Cassabdra、MongoDB 等 schema-less 的数据源中，有稀疏的行和宽的行。
  - Apache Drill 支持对不知道的 Schemas 的查询。
  - Drill 能自定义 Schema 或让系统自动发现它。记录的系统可以已经有 Schema 信息。不必管理 Schema 演进。
- Drill 支持 RDBMS、Hadoop 和 NoSQL。
- Drill 具有灵活和可扩展的体系结构。
  - 好的可扩展性体现在各种 API 和接口。

- SQL:2003 是基本语言，也能实施一个定制的解析器来支持 DSL（域专门的语言）、UDFs 和 UDTFs。
- 各种数据源和文件格式，能实施一个定制的 Scanner 来支持新的数据源和文件格式。
- 具有优化器：基于成本的优化器，定制的运算可以被实施，特殊的 Mahout 运算器（k-mean）正在被设计。运算器推送到数据源（RDBMS）。
  - 源查询→解析器 API。
  - 定制运算符、UDF→逻辑计划。
  - 服务树、CF、拓扑→物理计划/优化器。
  - 数据源和格式→Scanner API。

3.3.4 基于 PostgreSQL 的 SQL on Hadoop

Hadapt、EMC Greenplum HAWQ、Citus Data 是基于 PostgreSQL 的解决方案。其中，Hadapt 是专注 SQL on Hadoop 的厂商。Hadapt 解决方案的本质是数据在两种计算框架中分别存放，如图 3-46 所示，结构化数据存储于高性能关系型数据引擎，非结构化数据存储于 Hadoop 分布文件系统，对两种类型的数据交互依靠查询的切片执行。

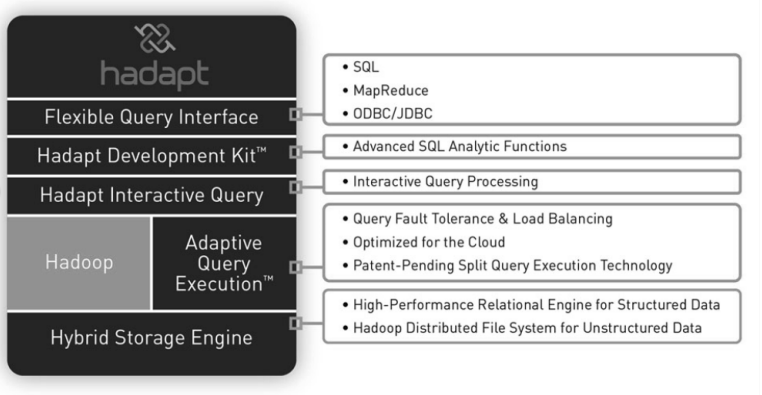


图 3-46 Hadapt 的体系架构

Hadapt 统一了关系型数据库环境和 Hadoop 环境。Hadapt 提供了一体化的分析环境，旨在对 Hadoop 里面的数据执行查询分析操作，还能对 SQL 环境中传统的结构化数据进行

查询分析。Hadapt 公司表示，通常采用的方法是使用由扩充型连接件联系起来的两个不同系统，但是这带来了延迟，因而导致这种方法显得很孤立。Hadapt 的平台设计成为可以在私有云或公共云环境上运行，提供了从一个环境就能访问所有数据的优点，所以除了 MapReduce 流程和大数据分析工具外，现有的基于 SQL 的工具也可以使用。Hadapt 可以在 Hadoop 层和关系型数据库层之间自动划分查询执行任务，提供了 Hadapt 所谓的优化环境，这种环境可以充分利用 Hadoop 的可扩展性和关系型数据库技术的快速度[25]。

3.4　大数据高级分析和可视化技术

3.4.1　传统数据仓库与联机分析处理技术

数据仓库是将不同数据源的数据进行 ETL 处理，形成统一的数据仓库，并基于数据仓库进行查询、联机分析处理（OLAP）等。数据批量 ETL 处理抽取架构由抽取和清洗、转换和加载几个主要部分组成。同时，批处理抽取架构还应提供缓存与通用数据处理服务。传统数据仓库 ETL 操作如图 3-47 所示。

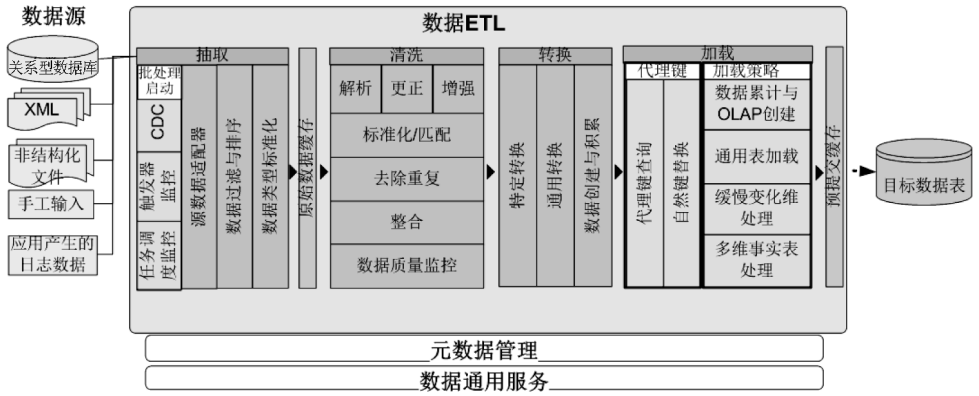


图 3-47　传统数据仓库 ETL 操作

详细的源数据在被处理载入数据仓库前，都将以符合三级范式的形式按照数据主题模型存储在 ODS（即 Operating Data Store，操作型数据存储）中。ODS 按功能分为数据缓冲

[25] http://www.hadapt.com。

区和统一信息视图区两个区域。数据通过应用集成平台（EAI）或 ETL 平台从各自的业务系统加载到 ODS 的缓冲区。

由于存放在 ODS 缓冲区的业务系统源数据或多或少存在着质量问题，如数据缺失、数据错误、不真实等问题，因此需要通过 ETL 技术，进行合并、映射、清洗、加载到数据仓库中，数据仓库的数据按照主题组织和存放，数据模型满足第三范式。如图 3-48 所示为数据仓库和数据分析总体框架。

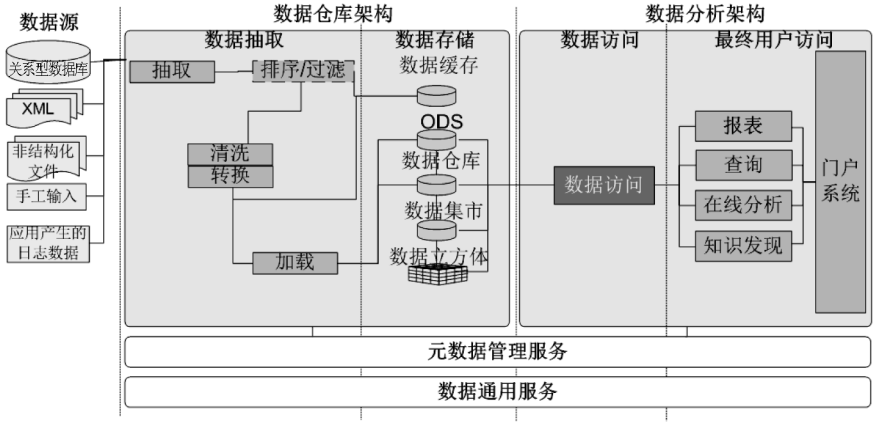


图 3-48　数据仓库和数据分析总体框架

数据集市是一组特定的、针对某个主题域、部门或用户分类的数据集合。这些数据需要针对用户的快速访问和数据输出进行优化，优化的方式可以通过对数据结构进行汇总和索引。

数据集市的数据按分析主题进行组织和存放，数据模型是星型结构的。而数据仓库的数据按照主题组织和存放，数据模型满足第三范式。从数据仓库到数据集市的数据移动应重点考虑从规范化建模到多维建模的映射，包括实体表向事实表、维表的映射，以及实体间关系到多维关系的映射。由于数据集市内需要按照地区、日期、类别等维度对数据进行汇总，因此数据仓库中所存放的数据需要经过累计汇总转换后才能加载到数据集市中。

联机分析处理（OLAP）包含多维观察、数据钻取、数据立方体（CUBE）运算等步骤。在数据仓库中，OLAP 分析所需的原始数据量是非常庞大的。一个分析模型，往往会涉及数百万、数千万条数据，甚至更多；而分析模型中包含多维数据，这些维又可以由浏览者做任意的提取组合。在 OLAP 中，大量的实时运算会导致时间的延滞。例如，一个 1000

万条记录的分析模型，如果一次提取 4 个维度进行组合分析，那么实际的运算次数将达到 4 的 1000 次方的数量。这样的运算量将可能导致数十分钟乃至更长的等待时间。如果用户对维度组合次序进行调整，或增加、减少某些维度的话，又将是一个重新的计算过程。解决 OLAP 运算效率问题涉及 OLAP 中一个非常重要的技术——数据立方体预运算。一个 OLAP 模型中，度量数据和维度数据应该事先确定，一旦两者确定下来，就可以对数据进行预先的处理。在正式发布之前，将数据根据维度进行最大限度的聚类运算，运算中会考虑到各种维度组合情况，运算结果将生成一个数据立方体，并保存在服务器上。

数据分析的角度包括报表、即席查询、联机分析和知识发现，如表 3-17 所示。

表 3-17　数据分析的 4 个方面

名　称	功　能	描　述	分析场景
报表	实现预定义和用户自定义报表功能	通过报表工具实现预定义报表的自动生成和分发，并能够灵活地实现用户自定义报表功能	• 静态数据 • 预定义报表 • 受限数据交互
即席查询	进行准实时的业务查询	通常即席查询的功能会涉及准实时的业务信息，可以由 ODS 区提供此类应用，通过即席查询，不需要非常专业的 SQL 知识即可完成信息的即席查看	• 事实发现 • 查询 • 报表
联机分析	利用 OLAP 分析手段实现多维度的交叉分析	利用 OLAP 分析工具，配合设计良好的 OLAP 数据模型，可以完成业务人员对业务的分析需求。联机分析的手段包括各种图形和表格的表现，以及在其上进行的多维度的交叉分析，帮助用户快速定位和解决问题	• 多维分析 • 例外管理 • 问题发现 • What-if 分析
知识发现（KDD）	利用数据挖掘、统计建模等知识发现技术实现特定的分析专题	用户获取有用信息的能力体现了数据仓库系统的价值，通过数据挖掘等高级统计分析技术，企业能够将数据源中有价值的信息（知识）识别出来并建立模型，同时通过自动化或半自动化的工具进行分析。数据库中的知识发现（KDD）过程如图 3-49 所示	• 规则发现 • 方案验证 • 交互图表 • 方案识别 • 相关性分析 • 聚类分析

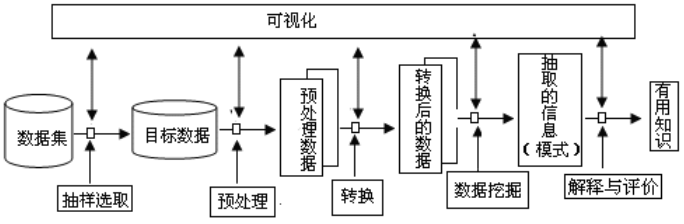


图 3-49　数据库中的知识发现（KDD）过程

3.4.2 大数据对传统分析的挑战

大数据对分析和可视化提出了更高的要求，要求数据分析从联机分析处理和报表向数据发现转变，从企业分析向大数据分析转变，从结构化数据向多结构化数据转变，要求分析和可视化能够支持对 PB 级以上的大数据进行分析，要求能够支持对关系型、非关系型、多结构化的、机器生成的数据做分析，要求能够支持重组数据成为新的复杂结构数据并进行分析和可视化，如，图分析、时间/路径分析（Time & Path Analysis），要求大数据的分析支持更快、更适应性的迭代分析，要求分析平台能够支持广泛的分析工具和编程方法，如 Python、R、C、C++、Java & SQL 等。

在大数据环境下，通过采样，可以把数据规模变小，以便利用现有的技术手段（关系数据库系统和数据仓库）进行数据管理和分析。然而在某些应用领域，采样将导致信息的丢失。商业组织积累了大量的交易历史信息，企业的各级管理人员希望从这些数据中分析出一些模式，以便从中发现商业机会，通过趋势分析，甚至预先发现一些正在涌现出来的机会。比如在金融服务行业，分析人员可以开发针对性的分析软件，对时间序列数据进行分析，寻找有利可图的交易模式，经过进一步验证之后，操作人员可以使用这些交易模式进行实际的交易，获得利润。但是，在全量数据上进行分析，意味着需要分析的数据量将急剧膨胀和增长。

在大数据环境下，典型的 OLAP 数据分析操作，如对数据进行聚集、汇总、切片和旋转等操作，已经不够用，还需要引入路径分析、时间序列分析、图分析、What-if 分析以及由于硬件/软件限制而未曾尝试过的复杂统计分析模型。例如，社交网络虚拟环境本质上是对实体连接性的描述。在社交网络中，每个独立的实体表示为图中的一个节点，实体之间的联系表示为一条边。通过社交网络图谱分析，可以从中发现一些有用的知识。比如发现某种类型的实体（有一种类型的实体把各个小组连接在一起，称为网络中的关键实体）。这些信息可以用于产品直销、组织和个体行为分析、潜在安全威胁分析等领域。

在大数据领域的行业应用需要在传统平台上扩展数据分析与统计功能。除了一般的分析功能还需要增加数据挖掘功能，例如 Teradata Aster 增加了很多新的分析功能，如统计、文本挖掘、图像、情感分析等。再如 IBM Netezza 则加入了对于 R 语言的支持，可以支持 R 的各类包，例如并行运算算法包、矩阵相关包。

3.4.3 大数据挖掘与高级分析

数据挖掘是从大量数据中寻找其规律的技术，是统计学、数据库技术和人工智能技术

的综合。数据挖掘是从数据中自动地抽取出模式、关联、变化、异常和有意义的结构。数据挖掘主要价值在于利用数据挖掘技术能发现规律并改善预测模型。

数据挖掘技术可以分为描述性技术和预测性技术，描述性技术了解数据中潜在的规律，预测性技术是用历史预测未来的技术。

数据挖掘的任务是从大量的数据中发现模式。根据数据挖掘的任务可分为多种类型，其中比较典型的有：关联分析、基于决策树或神经网络的分类分析、聚类分析、序列分析等，如图 3-50 所示是其中一些应用示例。下面分别介绍几种常见的数据挖掘类型。

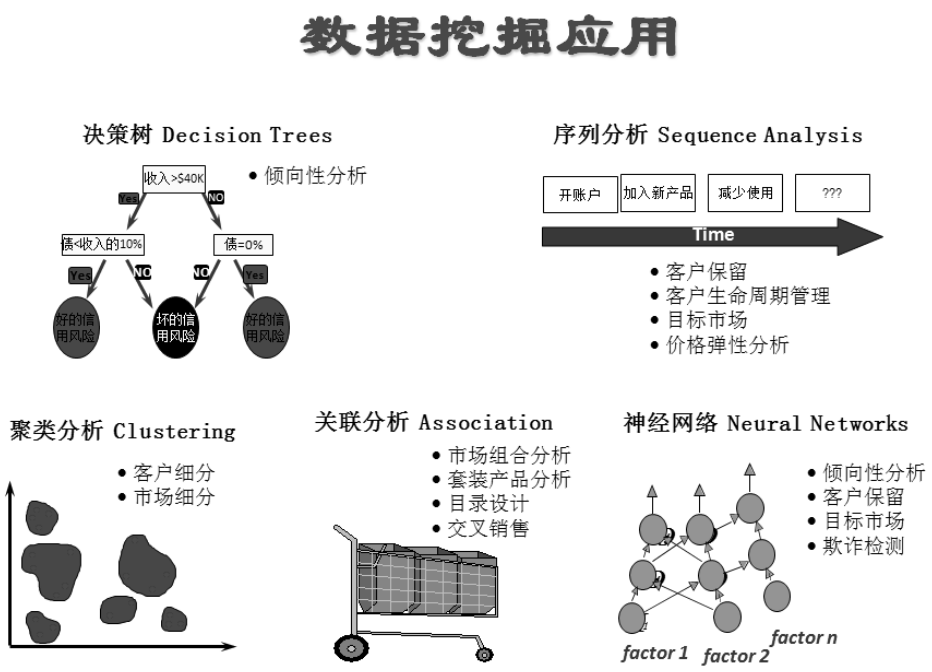


图 3-50　数据挖掘的各种应用

1. 关联（Association）分析

关联规则描述了一组数据项之间的关系。关联分析是在交易数据、关系数据或其他信息载体中，发现存在于项目集或对象集之间的关联规则，包括关联、相关性、因果结构或频繁出现的模式。在关联规则挖掘算法中，通常给出了置信度和支持度两个概念，对于置

信度和支持度均大于给定阈值的规则称为强规则，而关联分析主要就是对强规则的挖掘。关联规则模式属于描述型模式，发现关联规则的算法属于无监督学习的方法。关联分析广泛用于购物篮分析、交叉销售、商品目录设计等商业决策领域。沃尔玛就使用关联规则发现了哪些人同时购买了纸尿片和啤酒。例如，下面是在购物篮分析中的关联规则例子。

问题是：“什么商品组合，顾客可能会在一次购物中同时购买？”

购物篮分析：设全域为出售商店的集合（即项目全集），一次购物购买（即事务）的商品为项目全集的子集，若每种商品用一个布尔变量表示该商品的有无，则每个购物篮可用一个布尔向量表示。通过对布尔向量的分析，得到反映商品频繁关联或同时购买的购买模式。这些模式可用关联规则描述。

购买纸尿布与购买啤酒的关联规则可表示为：

```
diaper→beer [support=2%,confidence=60%]
```

support 为支持度，confidence 为置信度。

该规则表示：在所分析的全部事务中，有 2%的事务同时购买纸尿布和啤酒；在购买纸尿布的顾客中 60%也购买啤酒。

常用的关联分析算法有 Apriori 算法及它的各种改进或扩展算法。Apriori 算法是一种挖掘布尔关联规则频繁项集的算法。算法的核心思想是基于频集理论的一种递推方法，目的是从数据库中挖掘出那些支持度和信任度都不低于给定的最小支持度阈值和最小信任度阈值的关联规则。在这里，所有支持度大于最小支持度的项集称为频繁项集，简称频集。对于大规模、分布在不同站点上的数据库或数据仓库，关联规则的挖掘可以使用并行算法，如 Count 分布算法、Data 分布算法、Candidate 分布算法、智能 Data 分布算法（IDD）和 DMA 分布算法等。

2. 分类（Classification）分析

所谓分类是根据数据的特征为每个类别建立一个模型，根据数据的属性将数据分配到不同的组中。在实际应用过程中，分类规则可以分析分组中数据的各种属性，并找出数据的属性模型，从而确定哪些数据属于哪些组。这样就可以利用该模型来分析已有数据，并预测新数据将属于哪一个组。类的描述可以是显式的，如用一组特征概念描述；也可以是隐式的，如用一个数学公式或数学模型描述。

分类是事先定义好类别，属于有指导学习范畴。分类的目的是学会一个分类模型（称为分类器），该模型能把数据库中的数据项映射到给定类别中的某一个类中。要构造分类器，需要有一个训练样本数据集作为输入。训练集由一组数据库记录或元组构成，每个元组是一个由特征值组成的特征向量。此外，训练样本还有一个类别标记。一个具体样本的形式可表示为：

（v1,v2,...,vn; c）；其中 vi 表示特征值，c 表示类别。

常用分类算法有决策树、神经网络（NN）、贝叶斯分类（Bayes）等。决策树是一个树结构，它用样本的属性作为节点，用属性的取值作为分支。决策树的根节点是所有样本信息中信息量最大的属性，中间节点是以该节点为根的子树所包含的样本子集中信息量最大的属性，决策树的叶节点是样本的类别值。决策树学习是以实例为基础的归纳学习算法，它着眼于从一组无次序、无规则的事例中推理出决策树表示形式的分类规则。它采用自顶向下的递推方式，在决策树的内部节点进行属性值的比较并根据不同的属性值判断从该节点向下的分支，在决策树的叶节点得到结论。所以，从根节点到叶节点的一条路径就对应着一条合取规则，整棵决策树就对应着一组析取表达式规则。著名的决策树算法有 ID3 和改进的 C4.5。如图 3-51 所示为一个决策树的例子。

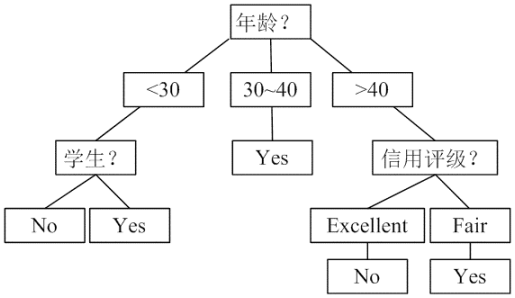


图 3-51　一个决策树的例子

神经网络（NN）算法是反映人脑结构及功能的一种数学模型，它是由大量的简单处理单元经广泛并行互联形成的一种网络系统，用以模拟人类进行知识的表示与存储以及利用知识进行推理的行为。它是对人脑系统的简化、抽象和模拟，具有人脑功能的许多特征。如图 3-52 所示为基于知识的神经网络的信息流程。

分类适合类别或分类体系已经确定的场合，目前分类分析已经成功地用于顾客分类、疾病分类、商业建模和信用卡分析等领域。

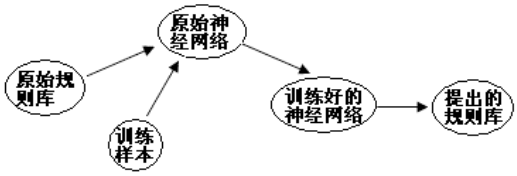


图 3-52　基于知识的神经网络的信息流程

3. 聚类分析（Clustering）

聚类是指一组彼此间非常“相似”的数据对象的集合。相似的程度可以通过距离函数来表示，由用户或专家指定。聚类分析是按照某种相近程度度量方法将数据分成互不相同的一些分组。每一个分组中的数据相近，不同分组之间的数据相差较大。好的聚类方法可以产生高质量的聚类，保证每一聚类内部的相似性很高，而各聚类之间的相似性很低。聚类分析的核心是将某些定性的相近程度测量方法转换成定量测试方法。采用聚类分析，系统可以根据部分数据发现规律，找出对全体数据的描述。

常用算法有 K 均值、最近邻、神经网络等。K 均值算法把 n 个对象根据它们的属性分为 k 个分割，$k < n$。它与处理混合正态分布的最大期望算法很相似，试图找到数据中自然聚类的中心。它假设对象属性来自于空间向量，并且目标是使各个群组内部的均方误差总和最小。

聚类分析用于很多领域，如对购物篮分析中，可用聚类分析基于其他人的兴趣来预测这个顾客的兴趣。

4. 序列（Sequence）分析

序列分析主要用于分析数据仓库中的某类与时间相关的数据，搜索类似的序列或子序列，并挖掘时序模式、周期性、趋势和偏离等。序列模式可以看成是一种特定的关联模型，它在关联模型中增加了时间属性。例如，它可以导出，“在两年前购买了福特轿车的顾客，有 70%可能在今年采取以旧换新的购车行动”，“在购买了自行车和购物篮的所有客户中，有 80%的客户会在两个月后购买打气筒”等。

5. 偏差检测（Deviation Detection）分析

用于检测并解释数据分类的偏差，即数据集中间显著不同于其他数据的对象。它有助

于滤掉知识发现引擎所抽取的无关信息，也可滤掉那些不合适的数据，同时可产生新的关注性事实。偏差包括很多有用的知识，如分类中的反常实例，模式的例外，观察结果对模型预测的偏差，量值随时间的变化等。偏差检测的基本方法是寻找观察结果与参照之间的差别，观察结果常常是某一个域的值或多个域值的汇总，参照是给定模型的预测、外界提供的标准或另一个观察。常用算法有决策树、神经网络、异常因子 LOF 检测等。常用应用有及时发现有欺诈嫌疑的异常行为等。

6. 预测模型（Predictive Modeling）分析

所谓预测即从数据库或数据仓库中已知的数据推测未知的数据或对象集中某些属性的值分布。建立预测模型的常用方法：回归分析、线性模型、支持矢量机、关联规则、决策树预测、遗传算法、神经网络等。后续章节将重点阐述预测分析。

7. 模式相似性挖掘

用于在时间数据库或空间数据库中搜索相似模式时，从所有对象中找出用户定义范围内的对象，或找出所有元素对中两者的距离小于用户定义的距离范围的元素对。模式相似性挖掘的方法有相似度测量法、遗传算法等。

3.4.4　大数据挖掘与高级分析库：Mahout

Apache Mahout 项目提供分布式机器学习和数据挖掘的库[26]。在被处理的数据很大时，Mahout 的目标是作为机器学习工具的一个选择。在当前的实现，这些可扩展的实施是用 Java 写的。很大一部分是建立在 Hadoop 分布式计算项目的基础上。这是一个 Java 库，它不提供用户接口、预打包的服务器或者安装程序，它只提供给开发者使用的一个工具框架，如图 3-53 所示。

[26] http://lucene.apache.org/mahout/。

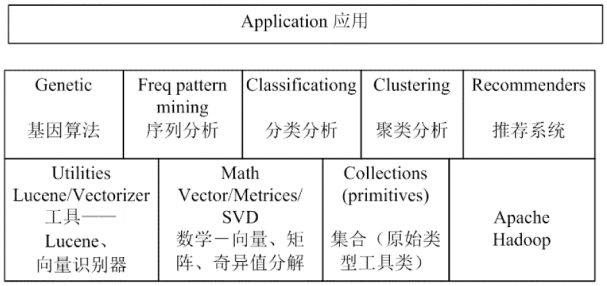


图 3-53　Mahout 算法库

3.4.5　非结构化复杂数据分析

1. 文本挖掘

文本挖掘是从文本数据中推导出模式。文本是无结构或者半结构化数据，它不同于结构化数据的挖掘，表 3-18 是数据挖掘和文本挖掘的区别。

表 3-18　数据挖掘与文本挖掘的区别

	数据挖掘	文本挖掘
研究对象	用数字表示的、结构化的数据	无结构或者半结构化的文本
对象结构	关系型数据库	自由开放的文本
目标	抽取知识，预测以后的状态	检索相关信息，提取意义，分类
方法	归纳学习、决策树、神经网络、粗糙集、遗传算法等	标引、概念抽取、语言学、本体
成熟度	从 1994 年开始得到了广泛应用	从 2000 年开始得到了应用

文本挖掘的过程[27]是通过文本分析、特征提取、模式分析的过程来实现的，主要技术包括：分词、文本结构分析、文本特征提取、文本检索、文本自动分类、文本自动聚类、话题检测与追踪、文本过滤、文本关联分析、文档自动摘要、信息抽取、智能问答、本体、XML 等半结构化文本挖掘、文本情感分析等。

搜索引擎是文本挖掘的重要领域，包括分类式和关键词索引式搜索引擎。分类式搜索

[27] 《文本挖掘技术》，杨建武，北京大学，研究生课程课件：http://wenku.baidu.com/view/7d7c3869a45177232f60a252。

引擎是将网络上的信息，包括网页、新闻组等按主题进行分类，由用户选择不同的主题来对网络上的信息进行过滤。关键词索引式搜索引擎的核心是一个关键词索引文件，该索引文件是一个倒排文件，倒排文件是一个已经排好序的关键词的列表，其中每个关键词指向一个倒排表，该表中记录了该关键词出现的文档集合以及在该文档中的出现位置。自动搜索引擎是能够自动获取网络上的信息，它们依靠爬虫程序在网络中不停地爬行和搜索，一旦发现新的信息，便自动对其进行分类，或用关键词对其进行索引，并将分类或索引结果加入到搜索引擎之中。智能搜索引擎在获取信息时要采用自动分类及自动索引等技术。这些技术均属于是自然语言处理和理解技术。如图 3-54 所示为文本挖掘结构模型。

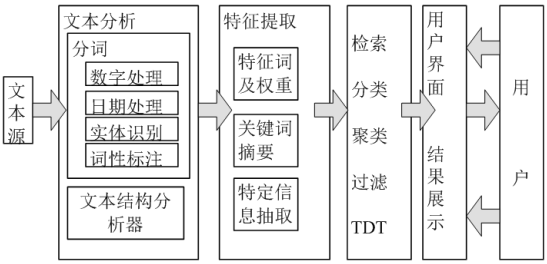


图 3-54　文本挖掘结构模型

2. Web 数据挖掘

在 Web 环境中，文档和对象一般都是通过链接由用户访问的，Web 数据挖掘可以采用文本挖掘的很多技术，但也有自身的特殊需求。Web 数据挖掘包括 Web 内容挖掘、Web 结构挖掘和 Web 使用模式挖掘等。如图 3-55 所示为 Web 分类挖掘。

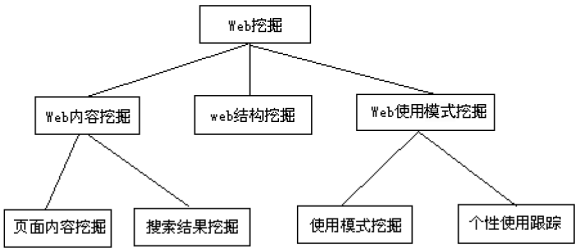


图 3-55　Web 挖掘分类

Web 内容挖掘是指在大量训练样本的基础上，得到数据对象之间的内在特征，并以此为依据进行有目的的信息筛选，从而获得指定内容的信息。Web 内容挖掘有两种策略：一

是直接挖掘文档的内容，如针对 Web 的查询语言 Weblog、WebOQL 等，利用启发式规则来寻找个人主页信息的 Ahoy 等；二是在工具搜索的基础上进行改进，主要是对搜索引擎的查询结果进行进一步的处理，得到更为精确和有用的信息，属于该类的有 WebSQL 及对搜索引擎的返回结果进行聚类的技术等。

Web 结构挖掘是挖掘 Web 的链接结构，并找出关于某一主题的权威网站。Web 结构挖掘是从 WWW 的组织结构和链接关系中挖掘知识。由于文档之间的互联，WWW 能够提供除文档内容之外的有用信息。利用这些信息，可以对页面进行排序，发现重要的页面。这方面工作的代表有 PageRank 和 CLEVER。此外，在多层次 Web 数据仓库中，也利用了页面的链接结构。

Web 使用模式挖掘是捕捉用户的存取模式或发现一个 Web 网站最频繁的访问路径，也称为 Web 路径挖掘（Path Analysis）。

3. 语音识别

语音识别[28]就是让机器通过识别和理解过程把语音信号转变为相应的文本或命令。语音识别技术主要包括特征提取、模式匹配准则及模型训练 3 个方面。在训练阶段，用户将词汇表中的每一词依次说一遍，并且将其特征矢量作为模板存入模板库。在识别阶段，将输入语音的特征矢量依次与模板库中的每个模板进行相似度比较，将相似度最高者作为识别结果输出，如图 3-56 所示。

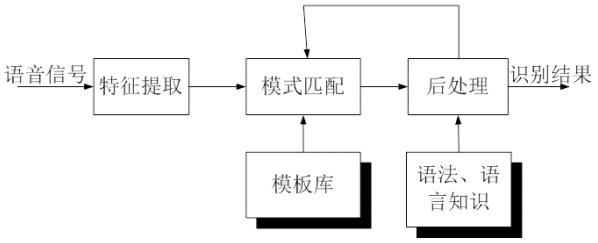


图 3-56　语音识别的过程

在语音识别的研究发展过程中，相关研究人员根据不同语言的发音特点，设计和制作了汉语（包括不同方言）、英语等各类语言的语音数据库，这些语音数据库可以为国内外

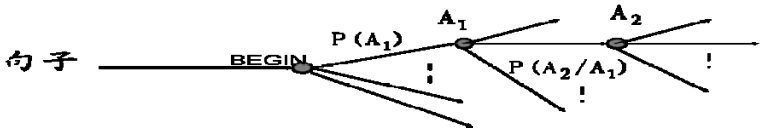
[28] http://baike.baidu.com/view/652891.htm。

有关的科研单位和大学进行汉语连续语音识别算法研究、系统设计及产业化工作提供充分、科学的训练语音样本。例如，MIT Media Lab Speech Dataset（麻省理工学院媒体实验室语音数据集）、Pitch and Voicing Estimates for Aurora 2（Aurora2 语音库的基因周期和声调估计）、Congressional Speech Data（国会语音数据）、Mandarin Speech Frame Data（普通话语音帧数据）、用于测试盲源分离算法的语音数据等。

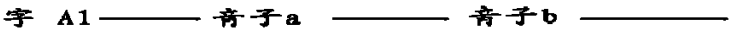
语音识别的主要方法是模式匹配法，主要算法包括基于模式匹配的动态时间规整（DTW）算法和基于统计模型的隐 Markov 模型（HMM）算法。

由于语音有较大的随机性，即使同一个人在不同时刻的同一句话发的同一个音，也不可能具有完全相同的时间长度，因此时间伸缩处理是必不可少的。动态时间规整（DTW）用满足一定条件的时间规整函数，描述待识别模式和参考模板的时间对应关系，求解两个模板匹配，结果就是累积距离测度最小对应的规整函数。

HMM 是一种随机过程，它用概率统计的方法来描述语音信号的变化过程。使用概率参数来进行估计和判决。用 HMM 实现连续语音识别的框架中，一段语音包括句法层、字层、语音层和声学层。在句法层，每个句子由若干字构成，每个字都选自于字库。

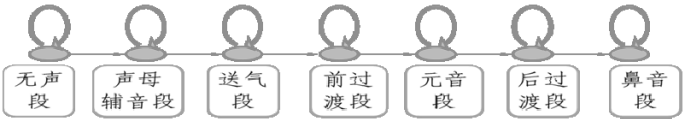


在字层，每个字由音子串接而成，需要一个数据库来描述每一个字如何用音子串接的。



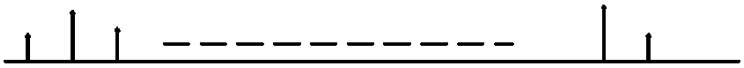
在语音层，每个音子用一个 HMM 模型及其相应的参数来描述（状态及其状态间的转移）。

音子 a：



发音音子的各个段构成相应的状态，HMM 是一个双重随机过程，基本单元发音速率（停留时间和转移时间）对应状态转移概率，声学变化（LPC 倒谱）对应输出序列，概率分布成混合高斯密度函数。

在声学层，提取语音帧特征矢量。



HMM 的状态是不确定或不可见的，只有通过观测序列的随机过程才能表现出来。观察到的事件与状态并不是一一对应的，而是通过一组概率分布相联系。HMM 模型参数是通过大量的训练数据进行统计运算而得到的，不仅可以用于特定人识别，而且可用于非特定人识别，这时，只要将大量不同人的多次发音用作训练数据即可。

在语音识别研究的前沿领域中，自然口语语音识别、人机口语对话系统、广播电视新闻自动记录系统等直接解决健壮性和灵活性问题的研究项目是研究热点。

4. 图像识别和分析

图像分析技术分类为 3 种基本范畴，低级处理包括图像获取、预处理，中级处理包括图像分割、表示与描述，高级处理包括图像识别、解释。

（1）图像识别

图像识别一般包括如下几个步骤，如图 3-57 所示。

图 3-57　图像识别的步骤

第 1 步是图像的获取，并建立颜色空间。颜色空间是对彩色的一种描述方法，它有很多种类型，如 RGB、CMY、YIQ、YUV、HSL 等。纹理是图像像素灰度级或颜色的某种变化，纹理分析主要研究如何获得图像纹理特征和结构的定量描述和解释，以便于图像分析、分割和理解。

第 2 步是图像的彩色边缘检测。边缘是图像的一个基本特征，携带了图像中的大量信息，边缘检测不仅能得到关于边界的有用的结构信息，而且还能极大地减少要处理的数据，很多图像处理和识别算法都以边缘检测为重要基础，检测的结果是要实现边缘的提取。

第 3 步是进行形态学处理，进行图像分割。首先要构造结构元素。利用构造的结构元素对图像进行膨胀操作。与原始图像相比，在边缘图像中存在一些细小的间隙，根据数学形态学原理，如果构造结构元素对图像进行膨胀操作，这些小间隙就会消失。膨胀运算后，

图像的边缘得到了很好的描述。然而，在目标物的内部，仍然存在一些空洞，可通过区域填充消除空洞。为了能够更加清楚地观察分割结果，对连通区域进行标记，并且用不同的颜色显示。提取出特定的连通区域，就能选择特定的对象。如图 3-58 所示为图像边缘检测的示例。

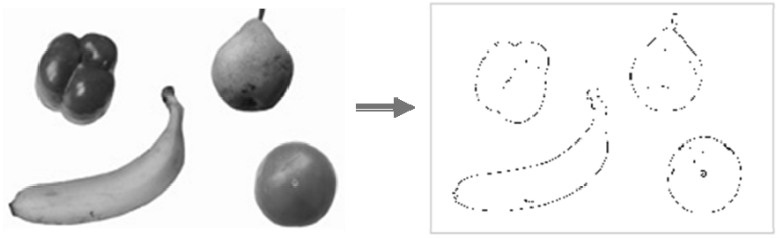


图 3-58　图像边缘检测（示例）

第 4 步是对图像的颜色特征和形状特征进行提取。图像经过边缘提取和图像分割等操作，就会得到边缘和区域，也就获得了目标的形状。在图像识别技术中，颜色是使用最广泛的特征之一，大部分系统都采用颜色比例分布作为颜色基本特征，这就是图像领域中的直方图法。任何物体的形状特征均可由其几何属性（如长度、面积、距离、凹凸等）、统计属性（如投影）和拓扑属性（如连通、欧拉数等）来进行描述。

第 5 步是图像的模式匹配。颜色特征提取后，如何用数值来有效地表示图像在颜色上的相似程度，这便是相似度量问题。相似度量也是直接影响识别效果的重要环节，在模式识别技术中，特征的相似度量均采用距离法，即特征的相似程度用特征向量的空间距离来表示。模式识别的许多方法，如决策理论、贝叶斯分类器、神经网络分类器、支持向量机等，都可以用到图像识别中来。如图 3-59 所示为图像模式匹配。

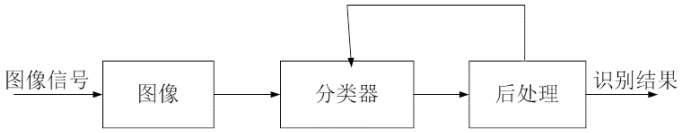


图 3-59　图像模式匹配

（2）图像检索与挖掘

对图像数据相似搜索，主要考虑两种图像标引和检索系统：（1）基于描述的检索系统，主要是在图像描述之上建立标引和执行对象检索，如关键字、标题、尺寸和创建时间等；（2）基于内容的检索系统，它支持基于图像内容的检索，如颜色构成、纹理、形状、对象

和小波变换等。

在基于内容的检索系统中，通常有两种查询：基于图像样本的查询（Image Sample-Based Queries）、图像特征描述查询（Image Feature Specification Queries）。基于图像样本的查询是指找出所有与给定图像样本相似的图像。其做法是把从样本中提取的特征向量（Feature Vector）（或特征标识（Signature））与已经提取并在图像数据中已经索引过的图像特征向量相比较。图像特征描述查询是指给出图像的特征描述或概括（如颜色、纹理或形状），把其转换为特征向量，与数据中已有的图像特征向量匹配，可以得到与样本图像近似的图像。主要方法有：基于颜色直方图的特征标识（Color Histogram-Based Signature）、多特征构成的特征标识（Multifeature Composed Signature）、基于小波的特征标识（Wavelet-Based Signature）、带有区域粒度的小波特征标识（Wavelet-Based Signature With Region-Based Granularity）。

为进行图像数据的多维分析，可以设计和构造出图像数据立方体。该数据立方体可包含针对图像信息的维度和度量，如颜色、纹理和形状。图像数据立方体对图像数据的多维分析是很有用的模型。然而，实现一个维度很大的数据立方体是极其困难的。在图像数据立方体中，我们要考虑颜色、方位、纹理、关键字等属性，而这其中很多属性是集合值而不是单值。如何设计出既能满足效率要求，又有足够表达能力的图像数据立方体是个亟待研究的问题。

分类、关联、预测分析等方法已经用于图像数据挖掘，目前图像分类主要采用决策树方法。图像数据中的关联规则挖掘，包括 3 类规则：图像内容与非图像内容的关联，与空间关系无关的图像内容的关联，与空间关系有关的图像内容的关联。要挖掘图像对象间的关系，可以把每一个图像看作一个事务，从中找出不同图像间出现频率高的模式。

5. 地理空间分析

空间数据挖掘技术作为当前数据库技术最活跃的分支与知识获取手段，在地理信息系统（GIS）中的应用推动着 GIS 朝智能化和集成化的方向发展。

空间数据挖掘（Spatial Data Mining-SDM），或者称为从空间数据库中发现知识，是为了解决空间数据海量特性而扩展的一个新的数据挖掘的研究分支，是指从空间数据库中提取隐含的、用户感兴趣的空间或非空间的模式和普遍特征的过程。空间数据库（数据仓库）中的空间数据除了其显式信息外，还具有丰富的隐含信息，如隐含了地质岩性与构造方面的信息；植物的种类是显式信息，但其中还隐含了气候的水平地带性和垂直地带性的信息。这些隐含的信息只有通过数据挖掘才能显示出来。常用的空间数据挖掘技术包括：序列分

析、分类分析、预测、聚类分析、关联规则分析、时间序列分析、粗集方法及云理论等。

3.4.6　实时预测分析

描述性分析使人们能够知道过去发生了什么，而预测性分析专注于正要发生什么，并进一步预测将来可能发生什么，如图 3-60 所示。随着新一代信息技术的发展，尤其是互联网、电子商务、社交网络、移动互联网、物联网等采集的数据量以及云计算、分布式计算、大数据、内存计算等经济高效的计算处理能力的增长，预测分析的重要性也在不断增长。实时预测分析是大数据分析的重要内容。

图 3-60　数据分析的框架

1. 预测模型

预测分析的基本原理是将采集的数据用于培训（或者学习）某种预测建模技术，就会生成预测模型，即：

数据 ＋ 预测建模技术 ＝ 预测模型

预测模型进而被推广用于预测。

预测模型简单来讲是一个数学函数，它能够获悉一组输入数据变量与目标变量之间的映射关系。学习分为监督式学习和非监督式学习。在监督式学习情况下，培训期间，会提

供给一个具有输入数据和期望输出或结果的预测模型。培训会反复进行，直到该模型获悉了给定输入与期望输出之间的映射关系为止。使用监督式学习的预测模型有反向神经网络、支持矢量机和决策树。预测模型也可使用非监督式学习。在这种情况下，仅为预测模型提供输入数据，然后预测模型的任务是确定不同的输入数据记录彼此之间的关联。聚类分析是最常用的使用非监督式学习的模型。

举一个客户流失分析和预测的例子[29]。现在要创建一个预测模型，该模型将能够确定企业的哪位客户最可能流失。首先，回到已经结构化的历史数据中，通过查看数据库，能够为现有客户和已流失的过去的客户编制一个相关特性的列表。该列表可能包含前 6 个月的投诉数量、前 4 个星期中公开的客户服务单数量、客户花钱购买商品或服务（在线或在店内）的频率和所花的金额，以及年龄、性别和人口统计等一般信息。表 3-19 是客户流失预测的例子，显示了两个这样的客户和所获取的其中每位客户的特性。客户 1 是一个现有客户且似乎很满意，但是客户 2 已流失。

表 3-19　客户流失预测的例子

客户 1	客户 2
过去 6 个月没有抱怨	过去 6 个月有 3 次抱怨
过去 4 个星期公开了 1 次客户服务单	过去 4 个星期公开了 2 次客户服务单
花了共 9786 美元来购买商品	花了共 1234 美元来购买商品
花了 987 美元用于服务	花了共 123 美元用于服务
过去 4 个星期购买了 12 件商品	过去 4 个星期购买了 2 件商品
54 岁	34 岁
男性	男性
生活在芝加哥	生活在洛杉矶

要创建一个预测模型，该模型将能够确定企业的哪位客户最可能流失。在监督式学习场景类型中，在培训期间将所有客户数据提供给一项预测技术，如神经网络或者决策树。在这种情况下，输入包括为每位客户准备的所有特性（满意度相关特性、人口统计等）以及相关结果，比表 3-17 数据分析的 4 个方面更大的数据集。该结果向预测模型表明，数据记录代表一位已流失还是未流失的客户。假设是该模型能够学习两个群体（现有的满意客户和已脱离的客户）之间的区别或模式。最终，数据和预测技术的结合将产生机器发现

[29] http://www.ibm.com/developerworks/cn/data/library/ba/ba-predictive-analytics1/。

的预测模型。

2. 预测分析的流程

创建一个完整的预测解决方案。首先是对问题明确定义，然后执行数据分析和预处理。之后将数据提交给一种预测技术以进行模型构建，进而评估模型的准确度。根据模型的准确度以及与预测错误相关的成本设定来鉴别阈值。之后，将业务决策与不同的阈值建立关联。最后，当将预测解决方案导出为一个 PMML（Predictive Model Markup Language，预测模型标记语言）文件时，即可对其进行部署和使用。完成这些步骤后，预测分析就真正履行了其承诺：从历史数据中学习有价值的模式并使用它们预测未来。

（1）预测数据的采集和预处理

预测的基础是数据。在大数据时代，数据正在呈现指数级增长。大数据平台能够帮助高效地完成海量数据的采集和处理，日志、传感器流和语言文本等非结构化数据经过 Hadoop 平台进行装载和高效处理，非结构化数据被转换为结构化数据，便于输入预测模型，如图 3-61 所示。

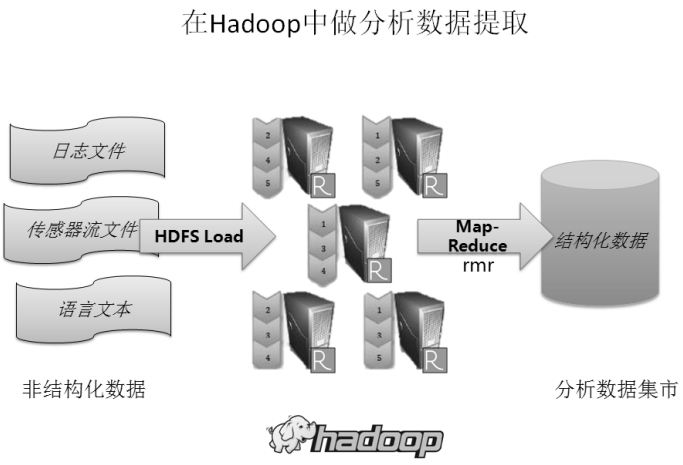


图 3-61　Hadoop 中做分析数据的提取

尽管可能会按原样使用某些数据字段，但是大部分字段需要进行某种预处理。历史数据的形式多种多样，例如客户流失分析的例子中，数据可能包含有关某位客户的结构化和非结构化信息。在这种场景下，结构化数据包含诸如客户年龄、性别、最近一月内的购买

次数等字段，这些信息是从客户账户记录和历史交易中获得的。非结构化数据可能包括相同客户针对所购买商品或服务提供的反馈意见或者评论。

数据预处理中，可以将某些输入字段的值放到一起。例如，所有年龄在 21 岁以下的客户都划分在学生类别中。类似地，所有 55 岁以上的客户将划分到退休人员类别中。年龄介于 21～55 之间的客户则属于工人的类别。很明显，这对最初的年龄字段进行了简化，然而这样做增加了将数据提交给某种预测技术以进行培训时的可预测性。实际上，数据预处理的最终目标就是增强输入字段的可预测性。

预处理的另一个目的是修改数据，从而使数据适合于培训。例如，根据模型构建使用的技术，可能需要对前面提到的 3 个与年龄有关的分类进行离散。对于这种情况，如果客户 A 是 25 岁，那么她的年龄将通过 3 个不同的字段表示：学生、工人和退休人员，分别映射为 0、1 和 0。出于同样的考虑，任何连续的字段都需要进行标准化。在本例中，最近一个月内的购买次数将转化为一个介于 0～1 之间的数。注意，原始字段本身已经是经过预处理后的结果，因为它表示一个聚合：某段时期内发生的所有交易的总购买次数。

此外，还可使用文本挖掘，在评论中找出损耗提示并创建一个损耗度量，也由 0～1 之间的一个值表示。使用客户的特性列表表示每个客户并将它们结合为一条记录。如果选择了 100 个特性并且存在 100000 名客户，那么最终数据集将包含 100000 个行（或记录）以及 100 个列。

数据是预测的基础。如果没有足够的数据，模型可能无法学习，这就很难培训出一个有意义的模型。某些预测模型的学习需要获取成千上万条记录，这正是大数据平台所擅长的。另一方面，数据的质量直接反映所学习出的模型的质量。在客户流失的例子中，这个阶段需要将一组表示某位客户的输入字段汇集为一条记录，该记录可能包含某些特性，如年龄、性别、邮政编码、最近 6 个月购买的商品数和退货的商品数，同时还包括一个目标变量，用于通知我们该客户在过去是否流失。然后，可以将一个客户记录通过数学方法描述为多维特性空间内的一个向量，这是因为需要使用多个特性来定义类型客户的对象。当将所有客户记录汇总到一块时，将成为包含数百万条记录的数据集。

要让数据处理变得更简单，除了允许用户按照以上描述操作数据外，还有许多统计包如 SAS、SPSS、R 可以使用户选择一个选项来自动预处理数据。例如，IBM SPSS Statistics 就允许自动构建特性。只需要选择“Transform menu > Prepare Data for Modeling”并单击“Automatic”即可。随后，您将被要求描述您的目标。可以从以下 4 个选项列表中选择。

这些选项是：1）平衡速度和准确度；2）优化速度；3）优化准确度；4）自定义分析。IBM SPSS Statistics 还允许对数据应用各种转换。这可以通过“Transform”菜单轻松访问。

（2）预测模型开发

模型培训可以从数据中获悉模式。在培训期间，所有数据记录都呈现出一种预测技术，该技术负责从数据中获悉模式。如果发生客户流失，数据中将包含可以区别流失客户和非流失客户的模式。注意，这里的目标是在输入数据（年龄、性别、最近一个月购买的商品数量等）和目标或因变量（流失和非流失）之间创建一个映射函数。

许多预测建模技术都可用于将这些大数据转换为洞察和价值，包括决策树、支持向量机（SVM）、神经网络（NN）、聚类、逻辑回归模型、关联规则等。不同的系统和供应商支持不同的技术，但是几种技术受到大多数商业或开源模型构建环境的支持。这些技术是通过学习大量历史数据中隐含的模式来实现这一点的。完成学习后，将生成一个预测模型。对模型进行验证后，就意味着该模型能够归纳所学习的知识并将归纳结果应用到新的情景中。

（3）模型验证和评估

在经历了模型构建过程后，预测模型被构建出来，如图 3-62 所示。接着，需要对模型进行验证，看预测模型是否有效；如果有效，看预测结果有多准确；如果模型有效且预测准确，并能够很好地推广，下面要做的事就是对该预测模型进行操作部署。

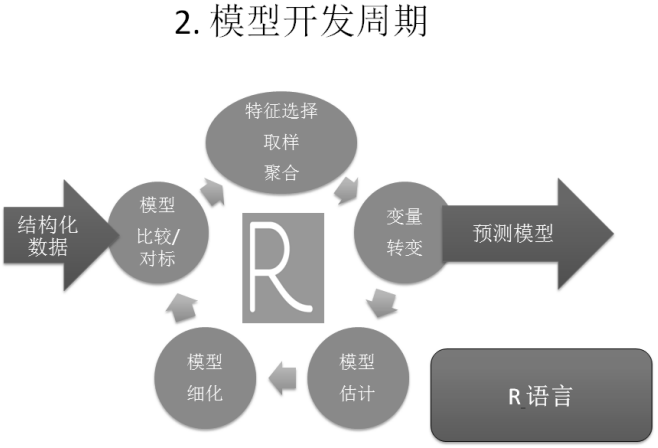


图 3-62　预测模型的开发过程

预留多达 30% 的数据用于模型验证，这是一种很好的做法。使用一个未用于模型培训的数据样本将允许对其精确度进行客观的评估。

当模型培训完成后，将提供有关模型健康状况的即时快照。这个快照让我们能够快速了解模型是否能够获悉目前涉及的任务。模型不需要达到 100% 的准确度。事实上，预测模型是因做出错误预测而为人所知。我们的目的是确保模型尽量减少错误，同时能够做出大量正确的预测。用预测分析的专业术语来说，您希望模型具有较低的假阳性（FP）率和假阴性（FN）率，这意味着具有高真阳性（TP）率和真阴性（TN）率。对于客户流失问题，如果模型为某位即将流失的客户准确指定了一个高流失风险，那么您将得到 TP。如果对某位实际上将继续保持高度忠诚的客户指定了一个低流失风险，那么您将得到 TN。然而，模型每次向某个满意客户指定高流失风险时，您将得到一个 FP。同样，如果模型向某个即将丢失的客户指定了一个低流失风险，那么您将得到一个 FN。

预测模型的原始输出通常为一个介于 0～1 或 1～-1 之间的值。但是，为了符合人们的阅读习惯，这个输出经常扩展为 0～1000 之间的值。模型的输出还可以根据给定的函数进行扩展，这样可以确保某个分数可以表示一个理想的 TP 和 FP 值。

后处理还意味着将评分嵌入到操作流程中。在这种情况下，需要将评分转换为业务决策。例如，对于客户流失问题，您的公司可能需要为高流失风险的客户提供更好的挽留计划。事实上，可以根据评分使用不同的鉴别阈值制定不同的业务决策，以及 TP、FP 及其成本方面的含义。

过去仅包含业务规则的决策管理解决方案如今也嵌入了预测模型。它们是通过将模型封装到规则中实现这一点的。通过结合业务规则和预测分析，企业能够从两种类型的知识中获益：专家知识和数据驱动的知识。在这种情况下，基于不同阈值的决策可以立即实现，甚至可以进行扩展以包含其他重要的决定性因素。例如，要判断某个客户的真正价值，该客户的流失风险评分可能还伴随提供了该客户在过去消费的金额。如果该金额巨大，同时流失风险也很高，那么说明阻止这名客户的流失将非常重要。通过结合专家知识和数据驱动知识，决策管理解决方案将变得更加智能，这是因为它们能够提供增强的决策功能。

（4）使用 PMML 实现大数据预测的有效部署

在预测模型操作部署阶段，PMML（Predictive Model Markup Language，预测模型标记语言）的标准能使预测模型在不同系统之间轻松迁移。借助 PMML，可使用 SAS、SPSS、R 等应用程序构建和验证一个预测模型，然后再将该模型保存为一个 PMML 文件，实现

模型平台无关的迁移。

预测模型标记语言 PMML 是由 Data Mining Group（DMG）提出，这是一个由多个公司组成的联盟，这些公司共同完成了对 PMML 的定义。所有商业的和开源的顶级统计和数据挖掘工具都支持 PMML。有了 PMML，预测解决方案迁移变得非常简单。PMML 允许企业和个人只使用一种语言来表示完整的预测解决方案，与开发解决方案的环境无关。从数据预处理和模型构建，一直到模型评分的后处理，PMML 能够在一个单一文件中呈现所有这些阶段。PMML 还可以呈现包含多个模型或一个模型组合的解决方案。

PMML 基于 XML，其模式遵循一个定义良好的结构，在这个结构中，元素和属性都用于定义以下内容。

- DataDictionary 元素，定义输入数据。
- MiningSchema 元素，定义用于处理策略的无效值、遗漏值和异常值。
- TransformationsDictionary 元素，定义数据预处理。
- NeuralNetwork、TreeModel、SupportVectorMachineModel、Scorecard 和 RegressionModel 等特定的模型元素定义各种建模技术。
- Output 元素定义模型输出的后处理。

PMML 还包含其他语言结构，包括用于模型验证、模型解释和评估的特定元素。PMML 可以使用一种清晰的结构化方式完整地呈现预测解决方案，PMML 为预测解决方案的交换定义了一个单一、清晰的流程。PMML 不仅成为了数据分析、模型构建和部署系统之间的桥梁，还成为了分析流程所涉及的所有人员和团队之间的桥梁。

PMML 的最新版本 V4.1 在 2011 年 12 月发布。然而，作为一种语言，PMML 已经有超过 10 年的历史，已经非常成熟和完善。

目前，市场上所有用于模型开发的统计工具都使用 PMML 导出模型，其中一些还提供导入功能，可以可视化模型并进行进一步的优化。一个值得一提的开源环境就是 KNIME，该模型导入和导出许多 PMML 模型；另一个就是用于统计计算的 R 语言项目。各种商用产品如 SAS、SPSS 也支持 PMML。

（5）实时预测分析的应用

预测分析在欺诈检测、医疗、产品推荐、故障诊断等领域有广泛的应用。很成功的预

测分析应用是欺诈检测。当客户每次刷信用卡或在线使用时，都有可能对其交易进行实时分析来检测出受到欺诈的可能性。预测分析在医疗保险行业有一些重要应用。通过知道哪些患者具有发生某种疾病的更高风险，可以制定预防措施来减轻风险，并最终挽救生命。互联网公司使用预测分析来推荐产品和服务。目前，已发展为从最喜爱的店铺和商家来预测优秀的影片、图书和歌曲推荐，还可以根据电子邮件、在线留言和搜索等内容，预测用户品味和偏好，推出有针对性的营销活动。其他应用专注于从传感器获取的数据。例如，可使用 GPS 移动设备数据来预测交通状况。随着这些系统变得越来越精确，将能够使用它们修改出行选择，例如，可在预计公路完全被汽车堵塞时乘坐地铁。此外，报告桥梁和建筑物等结构，以及能量转换器、水泵和气泵、闸门和阀门的当前状态的小型且经济高效的传感器的存在，也支持使用预测分析来维护和更改材料或流程，以预防发生欺诈和事故。通过支持构建预测维护模型，使用来自传感器的数据是帮助确保安全的一种明确方式。2010 年墨西哥湾的溢油灾难和 2007 年 I-35W 密西西比河大桥的坍塌，仅是在部署了传感器和预测维护模型后能够预防的重大事故的两个示例。

3.4.7 开源可视化工具：R 语言

1. R 语言概述

针对传统分析软件扩展性差以及 Hadoop 分析功能薄弱的特点，研究人员致力于对 R 和 Hadoop 的集成。R 是开源的统计分析软件，通过 R 和 Hadoop 的深度集成，把计算推向数据并且并行处理，使 Hadoop 获得了强大的深度分析能力。Purdue 大学的 RHIPE 项目（http://ml.stat.purdue.edu/rhipe/index.html）也致力于 R 和 Hadoop 的集成，为大数据分析提供开发环境的支持。Wegener 等人则实现了 Weka（类似于 R 的开源的机器学习和数据挖掘工具软件）和 MapReduce 的集成。

R 语言是从 S 统计绘图语言演变而来的，可看作 S 的“方言”。S 语言 20 世纪 70 年代诞生于贝尔实验室，由 Rick Becker、John Chambers、Allan Wilks 开发。基于 S 语言开发的商业软件 Splus，可以方便地编写函数、建立模型，具有良好的扩展性，在国外学术界应用很广。1995 年由新西兰 Auckland 大学统计系的 Robert Gentleman 和 Ross Ihaka，基于 S 语言的源代码，编写了一款能执行 S 语言的软件，并将该软件的源代码全部公开，这就是 R 软件，其命令统称为 R 语言。

R 是开源的统计绘图软件，也是一种脚本语言，有大量的程序包可以利用。R 中的向

量、列表、数组、函数等都是对象，可以方便地查询和引用，并进行条件筛选。R 具有精确控制的绘图功能，生成的图可以另存为多种格式。R 编写函数无须声明变量的类型，能利用循环、条件语句，控制程序的流程。

R 语言具有如下特点。

- 多领域的统计资源。目前在 R 网站上约有 2400 个程序包，涵盖了基础统计学、社会学、经济学、生态学、空间分析、系统发育分析、生物信息学等诸多方面。
- 跨平台。R 可在多种操作系统下运行，如 Windows、MacOS、多种 Linux 和 UNIX 等。
- 命令行驱动。R 即时解释，输入命令，即可获得相应的结果。
- 丰富的资源。涵盖了多种行业数据分析中几乎所有的方法。
- 良好的扩展性。十分方便地编写函数和程序包，跨平台，可以胜任复杂的数据分析、绘制精美的图形。
- 完备的帮助系统。每个函数都有统一格式的帮助，运行实例。
- GNU 软件。免费、软件本身及程序包的源代码公开。

R 语言的缺点。

- 用户需要对命令熟悉。与代码打交道，需要记住常用命令。
- 占用内存。所有的数据处理在内存中进行。
- 运行速度稍慢。即时编译，约相当于 C 语言的 1/20。
- 相比点击鼠标进行操作，R 仍能够大大提高效率。

R 语言与其他统计分析方法的比较如表 3-20 所示。

表 3-20　R 语言与其他统计分析方法的比较

R	速度快，简单易学，开源
SAS	速度快，有大量统计分析模块，但可扩展性略不足，价格较高
SPSS	复杂的用户图形界面，简单易学，但编程较困难
Splus	运行 S 语言，具有复杂的界面，与 R 完全兼容，价格较高

2. R 软件资源

R 语言官方网站[30]可以下载相关资源，The Comprehensive R Archive Network，简称 CRAN，是由世界几十个镜像网站组成的 R 资源网络，提供下载安装程序和相应软件包。各镜像更新频率一般为 1～2 天[31]，中国的镜像在中科院数学所[32]等网站。

R 程序包是多个函数的集合，具有详细的说明和示例。Windows 下的 R 程序包是经过编译的 zip 包。每个程序包包含 R 函数、数据、帮助文件、描述文件等。

R 程序包是 R 功能扩展，特定的分析功能，需要用相应的程序包实现。例如，系统发育分析，常用到 Ape 程序包，群落生态学 vegan 包等。R 语言的主要方法如表 3-21 所示。

表 3-21　R 语言的主要方法

ade4	利用欧几里得方法进行生态学数据分析
adephylo	系统进化数据挖掘与比较方法
Ape	系统发育与进化分析
apTreeshape	进化树分析
Boot	Bootstrap 检验
cluster	聚类分析
ecodist	生态学数据相异性分析
FD	功能多样性分析
geiger	物种形成速率与进化分析
Graphics	绘图
lattice	栅格图
maptools	空间对象的读取和处理
mefa	生态学和生物地理学多元数据处理
mgcv	广义加性模型相关
mvpart	多变量分解
nlme	线性及非线性混合效应模型

[30] http://www.r-project.org/。

[31] http://cran.r-project.org/。

[32] http://ftp.ctex.org/mirrors/CRAN/。

续表

ouch	系统发育比较
pgirmess	生态学数据分析
Phangorn	系统发育分析
picante	群落系统发育多样性分析
Raster	栅格数据分析与处理
Seqinr	DNA 序列分析
Sp	空间数据处理
spatstat	空间点格局分析，模型拟合与检验
splancs	空间与时空点格局分析
Stats	R 统计学包
SDMTools	物种分布模型工具
vegan	植物与植物群落的排序，生物多样性计算

CRAN 任务视图如表 3-22 所示。

表 3-22　CRAN 任务视图

贝　叶　斯	贝叶斯推理
化学物理	计量化学和计算物理
临床试验	临床试验的设计、监控和分析
集群	集群分析和有限混合模型（FMM）
分布	概率分布
经济学	计量经济学
环境计量学	生态和环境数据分析
试验设计	试验设计和试验分析
金融	实验金融
基因学	统计基因学
图	图显示、动态图、图设备可视化
gR	R 中的图模型
高性能计算	基于 R 的高性能和并行计算
机器学习	机器学习和统计学习

续表

贝　叶　斯	贝叶斯推理
医学影像	医学影像分析
多变量	多变量统计
自然语言理解	自然语言理解
优化	优化和数学编程

R commander 是 R 的图形界面之一，是 John Fox 教授编写的，适用于不希望 R 编程的用户。随着用户的操作，其窗口还可以显示出相应操作的 R 程序，对于初学者可能会有帮助。

3. R 语言函数

R 是一种解释性语言，输入后可直接给出结果。

（1）函数

R 语言功能靠函数实现。函数形式：

函数(输入数据, 参数=)

如果没有指定，则参数以默认值为准。例如：

平均值：mean(x, trim = 0, na.rm = FALSE, ...)

线性模型：lm(y～x, data=test)

每一个函数执行特定的功能，后面紧跟括号，例如：

平均值：mean()　　求和：sum()　　绘图：plot()　　排：sort()

除了基本的运算之外，R 的函数又分为”高级”和”低级”函数，高级函数可调用低级函数，这里的“高级”函数习惯上称为泛型函数。如 plot()就是泛型函数，可以根据数据的类型，调用底层的函数，应用相应的方法绘制相应的图形。这就是面向对象编程的思想。函数查询的方法是：Help>Html help>packages。

（2）元素

R 处理的所有数据、变量、函数和结果都以对象的形式保存。对象是由各元素组成的。

每个元素，都有自己的数据类型，主要的数据类型有：

- 数值型 Numeric，如 100、0、−4.335。
- 字符型 Character，如“China”。
- 逻辑型 Logical，如 TRUE、FALSE。
- 因子型 Factor，表示不同类别。
- 复数型 Complex，如 2 + 3i。

（3）对象的类

向量（vector），一系列元素的组合。如 c(1,2,3); c("a","a","b","b","c")。

因子（factor），因子是一个分类变量，c("a","a","b","b","c")。

矩阵（matrix），二维的数据表，是数组的一个特例。

```
x <- 1:12 ;   dim(x) <- c(3,4)
      [,1]  [,2]  [,3]  [,4]
[1,]    1     4     7    10
[2,]    2     5     8    11
[3,]    3     6     9    12
```

数组（array），数组是 *k* 维的数据表（k in 1:n，n 为正整数）。向量（n = 1），矩阵（n = 2），高维数组（n >= 3）。

数据框（dataframe），是由一个或几个向量和（或）因子构成的，它们必须是等长的，但可以是不同的数据类型。

列表（list），列表可以包含任何类型的对象，可以包含向量、矩阵、高维数组，也可以包含列表。

4. 运算符

数学运算，运算后给出数值结果，+、−、*、/、^（幂）。

比较运算，运算后给出判别结果（TRUE FALSE），>、<、<=、>=、==、!=。

逻辑运算，与、或、非、!、&、&&、|、||。

5. 数据表、数据框

每个数据表可以看作一个数据框（dataframe）。每一列（column）作为一个向量（vector）。由很多不同类型的向量组成，如字符型、因子型、数值型。每一行（row）作为一个记录（entry）。生成数据框有两种办法：一是从外部数据读取，二是各类型因子组合成数据框。

6. 数据读取

最为常用的数据读取方式是用 read.table() 函数或 read.csv()函数，读取外部 txt 或 csv 格式的文件。

- txt 文件，制表符间隔。
- csv 文件，逗号间隔。

一些 R 程序包（如 foreign）也提供了直接读取 Excel、SAS、DBF、MATLAB、SPSS、SYSTAT、Minitab 文件的函数。

一个例子，现有 6 名患者的身高和体重如下，用 R 语言检验体重除以身高的平方是否等于 22.5。

	1	2	3	4	5	6
身高 m	1.75	1.80	1.65	1.90	1.74	1.91
体重 kg	60	72	57	90	95	72

数据量较少时，数据读取可以从控制台直接输入：

```
height<-c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)
weight<-c(60, 72, 57, 90, 95, 72)
sq.height<-height^2
ratio<-weight/sq.height
t.test(ratio, mu=22.5)
```

数据量较大时用 read.table 函数从外部 txt 文件读取数据。

第 1 步，将 Excel 中的数据另存为 txt 格式（制表符间隔）或 csv 格式。第 2 步，用 read.table()或 read.csv()函数将数据读入 R 工作空间，并赋值给一个对象。一般从 txt 文档读取数据，每一行作为一个观测值，每一行的变量用制表符、空格或逗号间隔开。

```
read.table("位置", header=T)
read.csv("位置",header=T)
#从外部读取数据
data1<-read.table("d:/t.test.data.txt",header=T)
bmi<- data1$weight/data1$height^2
t.test(bmi, mu=22.5)   #t 检验
```

7. 脚本语言

脚本语言语法和结构通常比较简单，不需要编译，通过解释器对脚本进行解释，从而给出结果，能用简单的代码完成复杂的功能，但是速度较慢。常见的脚本语言有 Windows 批处理程序、PHP、Perl、Python、Ruby、JavaScript 等。R 自带的脚本编辑器 Editplus、TinnR、Ultraedit、Emacs、Notepad++ 与 NpptoR 组合等。

8. 绘图

R 语言的绘图分析，包括点图、饼图、趋势图、正态分布图、拓扑图等。R 语言具备卓越的绘图功能，通过参数设置对图形进行精确控制。绘制的图形能满足出版印刷的要求，可以输出 jpg、tiff、eps、emf、pdf、png 等各种格式。通过与 GhostScript 软件的结合，可以生成 600dpi、1200dpi 等各种分辨率和尺寸的图形。如图 3-63 所示为 R 图例。

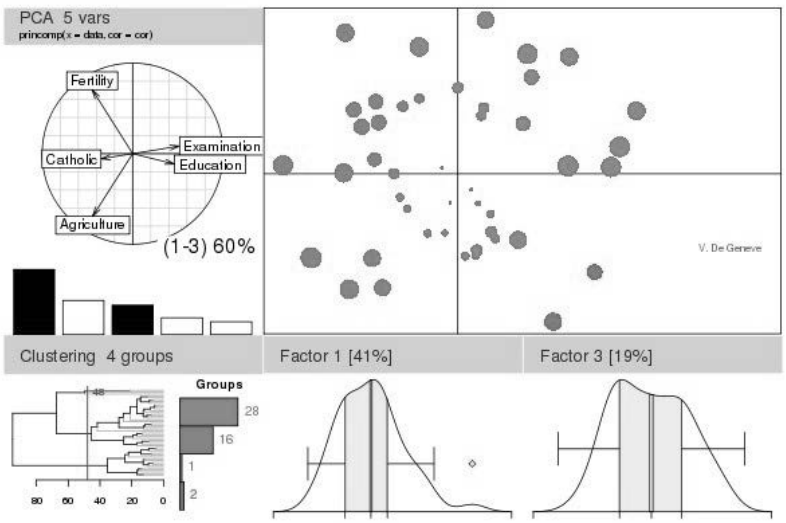


图 3-63　R 图例

绘图是通过绘图函数结合相应的参数完成的。绘图函数包括高级绘图函数和低级绘图函数，如表 3-23 和表 3-24 所示。

表 3-23　高级绘图函数

plot()	绘制散点图等多种图形，根据数据的类，调用相应的函数绘图
hist()	频率直方图
boxplot()	箱线图
stripchart()	点图
barplot()	柱状图
dotplot()	点图
piechart()	饼图
matplot()	数学图形

表 3-24　低级绘图函数

lines()	添加线
curve()	添加曲线
abline()	添加给定斜率的线
points()	添加点
segments()	折线
arrows()	箭头
axis()	坐标轴
box()	外框
title()	标题
text()	文字
mtext()	图边文字

R 可以灵活地编写程序，用户自己编写的程序可以直接调用。编程时无须声明变量的类型，这与 C、C++等语言不同。

3.4.8　可视化技术

可视化技术分为两类，一是可视化报表，二是可视化分析。可视化报表用图和表来描

述业务绩效，通常通过度量和时间系列信息来定义。可视化报表的主要类型是一个仪表盘或者一个记分卡，给绩效一个可视化的快照。最好的仪表盘和记分卡能使得用户钻取一个或更多的层次来获得关于被度量对象的更详细的信息。本质上，仪表盘是一个可视化的异常报告，用可视化技术来强调绩效的异常。主要可视化报表工具有仪表盘、报告、基于 Excel 的分析、维度分析（OLAP 分析）。

可视化分析，使得用户可视化地探索数据来发现新的洞察。当可视化报表结构化了关于预定义度量对象的数据的方向时，可视化分析提供一个更高程度的数据互动。用可视化分析，用户能够按照思维的速度可视化地过滤、比较和关联数据。可视化分析工具经常也会组合预报、建模和统计分析、What-if、预测分析等方法。与 OLAP 工具相比，可视化分析工具不需要 IT 人员设计一个多维数据模型，他们用一个“即时装载和处理”的方法，直接把多个源头的原始数据装载进来，通过他们的公共关键词简单链接到表中，并得到一个数据集的统一的视图。可视化分析通常提供一个 point-and-click 接口，借此分析能被通过交互的统计图来增强。这使得更快地分析、更好地决策和更有效地展现和理解结果。即点击饼图或者柱图，相关分析结果可以进一步展现。

如今，学习数据可视化的渠道有很多，2012 年 Netmagzine 列举了二十大数据可视化工具[33]，无论是准备制作简单的图表还是复杂的图谱或者信息图，这些工具都能满足需要。而且，这些工具大多数是免费的。

1. 入门级工具

Excel 的图形化功能并不强大，但作为一个入门级工具，Excel 是快速分析数据的理想工具，也能创建供内部使用的数据图。CSV（逗号分隔值）和 JSON（JavaScript 对象注释）并不是真正的可视化工具，但却是常见的数据格式。必须理解它们的结构，并懂得如何从这些文件中导入或者导出数据。几乎所有数据可视化工具都支持 CSV、JSON 中至少一种格式。

2. 在线数据可视化工具

Google Chart API[34]工具集提供动态图表工具，功能很丰富，能够在所有支持

[33] Netmaganize 数据可视化工具：http://www.netmagazine.com/features/top-20-data-visualisation-tools。

[34] https://developers.google.com/chart/。

SVG\Canvas 和 VML 的浏览器中使用，但是 Google Chart 图表在客户端生成，对那些不支持 JavaScript 的设备将无法使用，此外也无法离线使用或将结果另存其他格式，如图 3-64 所示为几种 Google Chart。Flot[35]是一个优秀的线框图表库，是一个纯 JavaScript 绘画库，它能基于 jQuery 开发图表，能够在客户端根据任何数据集快速生成图片，支持所有支持 Canvas 的浏览器（目前主流的浏览器如火狐、IE、Chrome 等都支持）。Raphaël[36]是创建图表和图形的 JavaScript 库，与其他库最大的不同是输出格式仅限 SVG 和 VML，如图 3-65 所示为 Raphaël 的动态图形库。SVG 是矢量格式，在任何分辨率下的显示效果都很好。D3[37]（Data Driven Documents）是支持 SVG 渲染的另一种 JavaScript 库，能够提供大量线性图和条形图之外的复杂图表样式，例如 Voronoi 图、树形图、圆形聚类和单词云等，如图 3-66 所示为 D3 气泡图。Visual.ly[38]是做信息图的首选工具，提供了大量信息图模板，如图 3-67 所示。

图 3-64　Google Chart

[35] http://www.flotcharts.org/。

[36] http://raphaeljs.com/。

[37] http://d3js.org/。

[38] http://visual.ly/。

图 3-65　Raphaël 的动态图形库

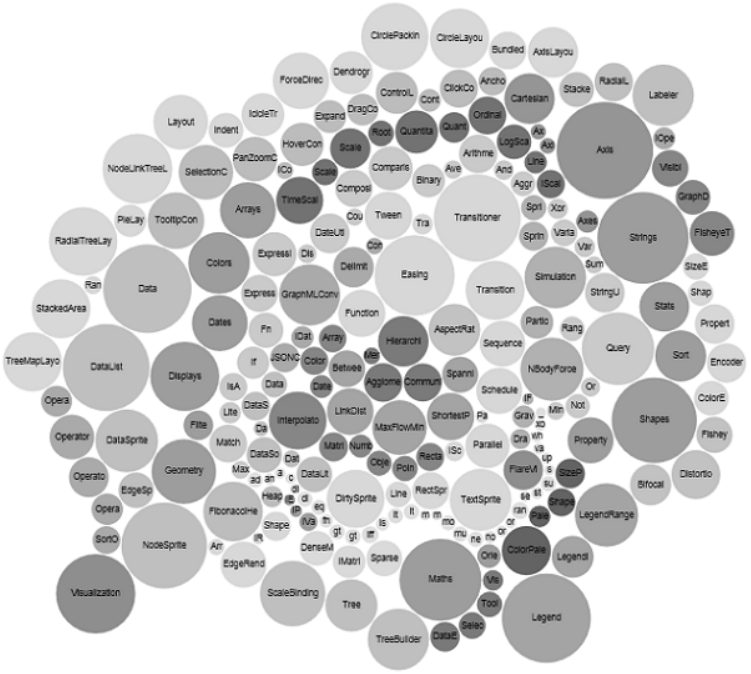


图 3-66　D3 气泡图

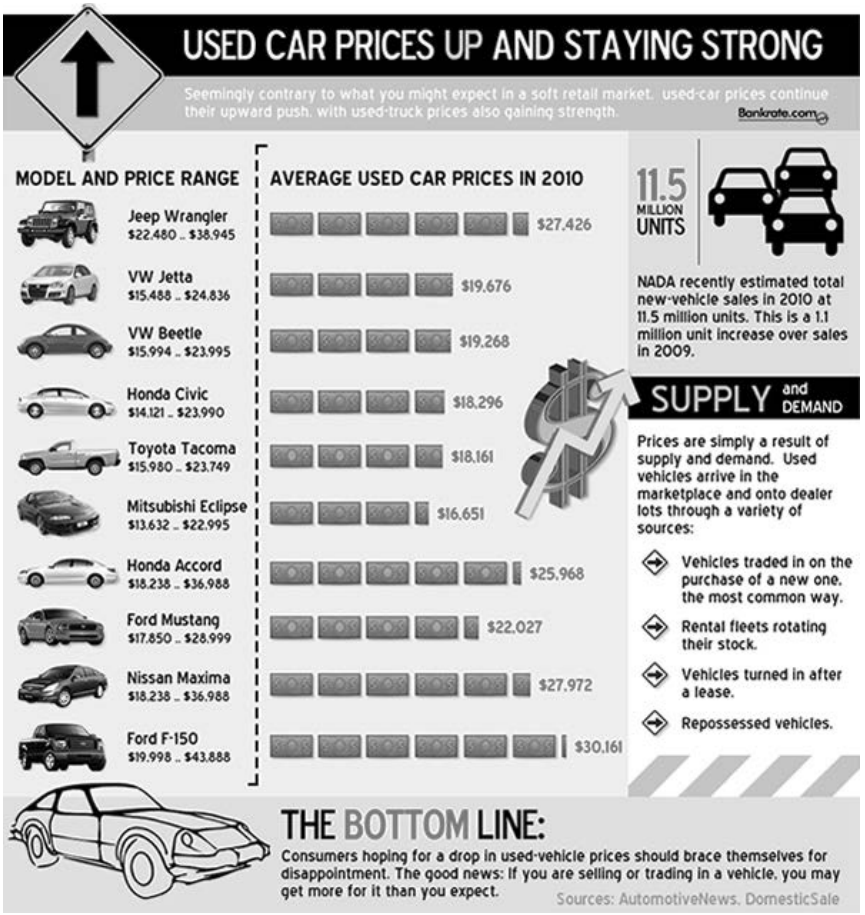


图 3-67　Visual.ly 信息图

3. 互动图形用户界面（GUI）控制

随着在线数据可视化的发展，可视化的互动性进一步增强，按钮、下拉列表和滑块都在进化成更加复杂的界面元素，例如能够调整数据范围的互动图形元素，推拉这些图形元素时输入参数和输出结果数据会同步改变，在这种情况下，图形控制和内容已经合为一体。

为方便客户浏览数据，Crossfilter[39]能够创建出既是图表，又是互动图形用户界面的

[39] http://square.github.io/crossfilter/。

JavaScript 小程序。例如，当你用鼠标点击图表中的不同时间，关联的其他 3 个图表的数据也会随之改变。如图 3-68 所示为 Crossfilter 的互动 GUI。JavaScript 库 Tangle[40]进一步模糊了内容与控制之间的界限，在 Tangle 生成的方程、数据表或者一句话中，读者可以调整输入值获得不同的结果数据。

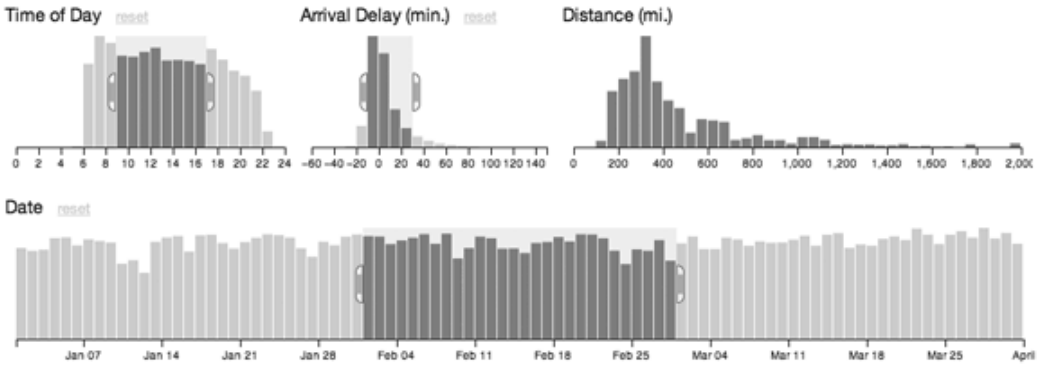


图 3-68　Crossfilter 的互动 GUI

4. 地图工具

地图生成是 Web 上最困难的任务之一。Google Maps 的出现完全颠覆了过去人们对在线地图功能的认识。而 Google 发布的 Maps API 则让所有的开发者都能在自己的网站中植入地图功能。近年来，在线地图的市场成熟了很多，如果你需要在数据可视化项目中植入定制化的地图方案，目前市场上已经有很多选择，但是知道在何时选择何种地图方案则成了一个很关键的业务决策。

Modest Maps[41]是一个很小的地图库，只有 10KB 大小，是目前最小的可用地图库。Leaflet[42]，是另外一个小型化的地图框架，通过小型化和轻量化来满足移动应用网页的需要。Leaflet 和 Modest Maps 都是开源项目，有强大的社区支持，是在网站中整合地图应用的理想选择。Polymaps[43]也是一个地图库，它是一个基于 SVG 的图像和矢量平铺地图

[40] http://worrydream.com/Tangle/。

[41] http://modestmaps.com/。

[42] http://leaflet.cloudmade.com/。

[43] http://polymaps.org/。

的 JavaScript 库，主要面向数据可视化用户，如图 3-69 所示。Polymaps 在地图风格化方面有独到之处，类似 CSS 样式表的选择器。OpenLayers[44]是所有 Web 地图库中可靠性最高的一个。

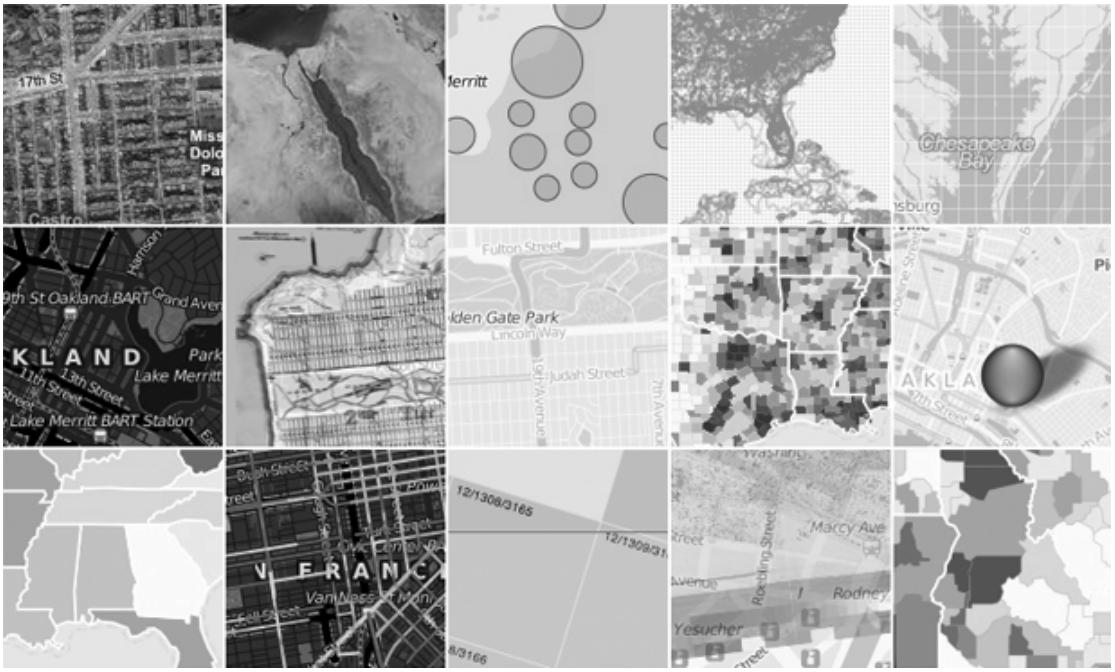


图 3-69　Polymaps 地图库

Kartograph[45]的标记线是对地图绘制的重新思考，非莫卡托投影的地图，如图 3-70 所示为 Kartograph 地理库。如果不需要调用全球数据，而仅仅是生成某一区域的地图，Kartograph 很有优势。CartoDB[46]是地图可视化和分析工作，可以很轻易就把表格数据和地图关联起来，例如输入 CSV 通信地址文件，CartoDB 能将地址字符串自动转化成经度、维度数据并在地图上标记出来。

[44] http://openlayers.org/。

[45] http://kartograph.org/。

[46] http://cartodb.com/。

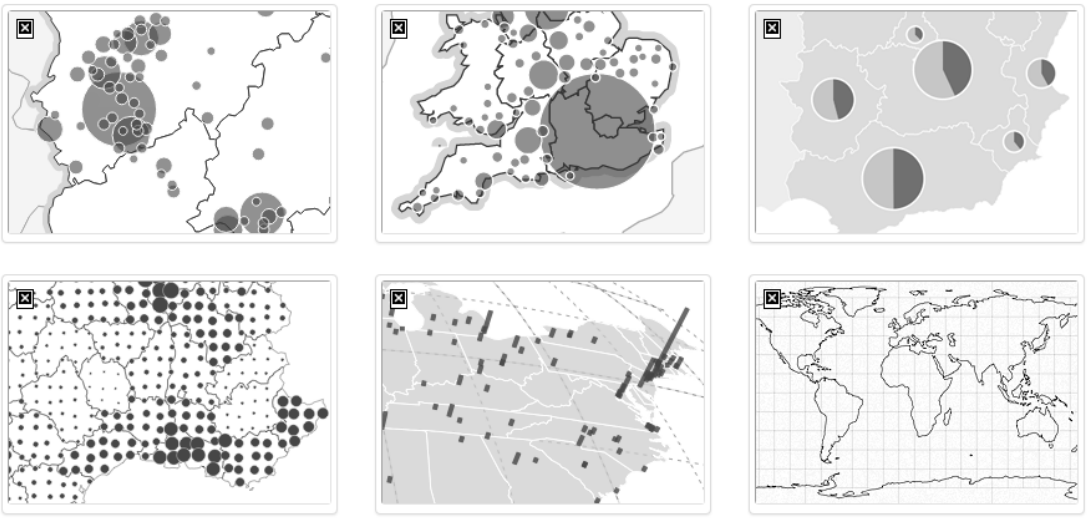


图 3-70　Kartograph 地图库

5. 可视化设计工具

可视化设计，更多地采用了桌面应用和编程环境。Processing[47]是一个很专业的可视化设计应用，可以在几乎所有平台上运行，用以产生图像、动画和交互，如图 3-71 所示为 Processing 可视化图。可视化工作只需要编写一些简单的代码，然后编译成 Java。目前，还有一个 Processing.js 项目，可以让网站在没有 Java Applets 的情况下更容易地使用 Processing。由于端口支持 Objective-C，也可以在 iOS 上使用 Processing。经过数年发展，Processing 社区目前已近拥有大量可视化实例和代码。NodeBox[48]是 OS X 上创建二维图形和可视化的应用程序。它是使用 Python 语言的程序，与 Processing 类似，但是没有互动功能。

6. 专家级可视化分析工具

专业数据分析师和数据科学家更为精通专业数据分析工具。IBM 的 SPSS 和 SAS 是数据分析行业的商业化标准工具，大型企业组织和学术机构一般会使用这些工具。Netmagnize 的文章中也介绍了几个常用的开源的专家级工具，这些工具都有强大的社区支

[47] http://www.processing.org/。

[48] http://nodebox.net/。

持，性能很好，插件的支持也不错。

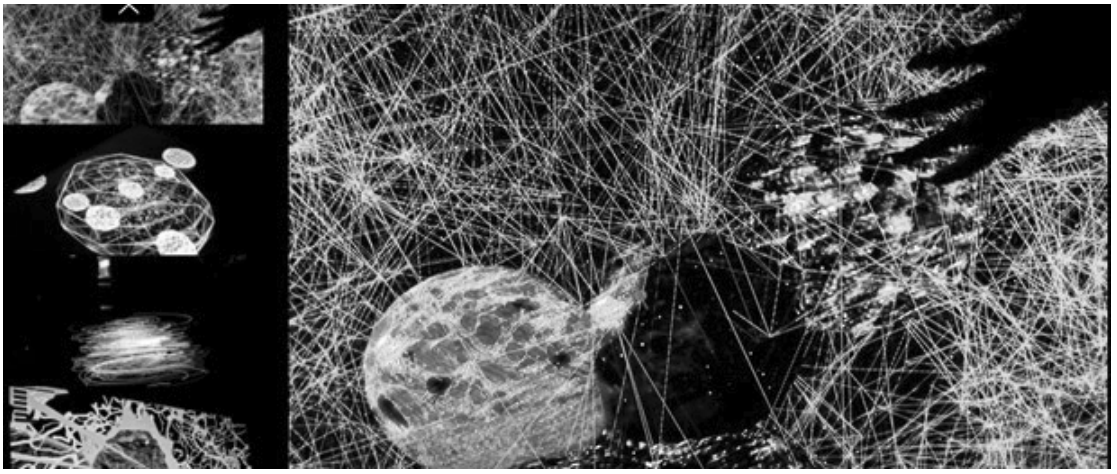


图 3-71　Processing 可视化图

作为用来分析大数据集的统计组件包，R 需要较长的学习实践。但是 R 拥有强大的社区和组件库，而且还在不断成长。上面一节，已经对 R 语言做了重点介绍，R 是开源大数据平台上的理想的分析和可视化工具。

Weka[49]是一个能根据各种特征做分类分析和聚类分析的数据挖掘软件工具，它拥有强大的做数据挖掘的机器学习算法集合，同时也能生成可视化的简单图表。

Gephi[50]是进行社交图谱数据可视化分析的工具，它能描绘出相对于网络中的其他节点，两个节点之间是如何相关联的。它不但能处理大规模数据集并生成漂亮的可视化图形，还能对数据进行清洗和分类。图 3-72 表示了正在做分析的 Gephi 图形，在原彩色图中不同颜色的区域表示数据的聚类是相似的。

[49] http://www.cs.waikato.ac.nz/ml/weka/。

[50] http://gephi.org/。

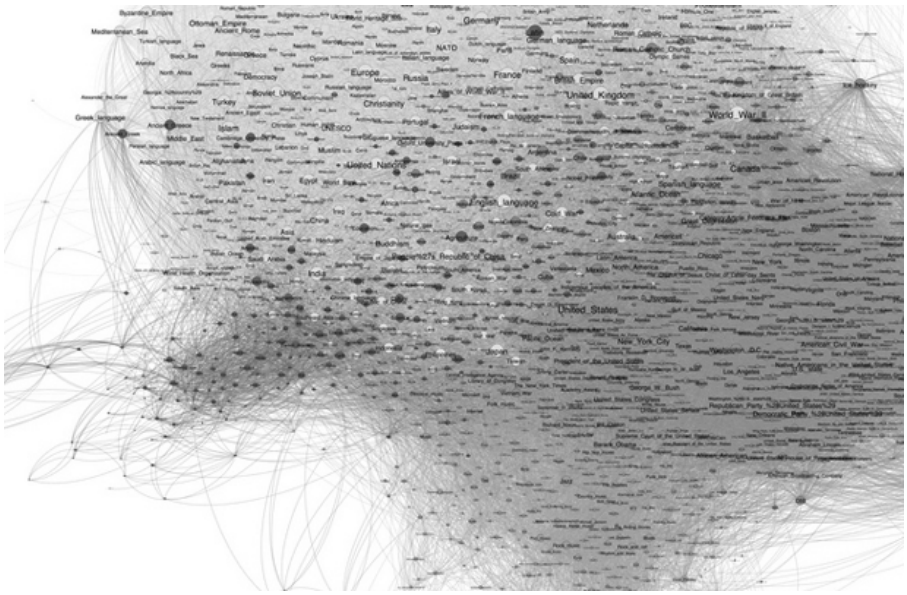


图 3-72　Gephi 可视化分析工具

3.5　以银行客户分析为例的大数据的技术环境部署

3.5.1　银行客户大数据应用体系架构

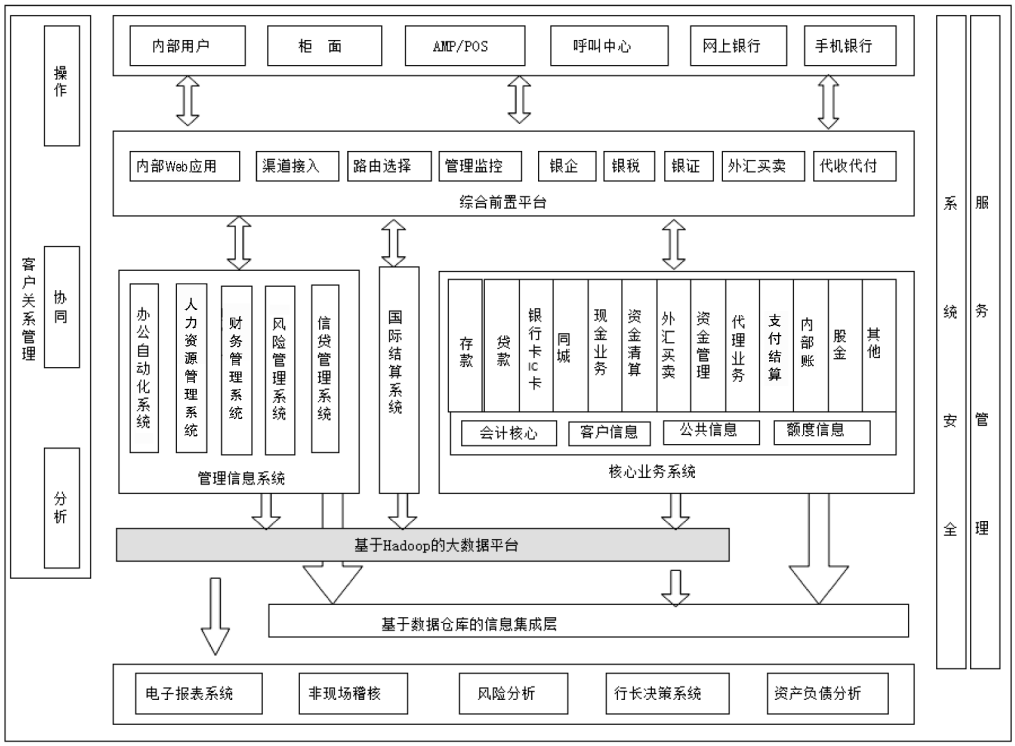
银行客户分析的案例中，企业将引入 Hadoop 为基础的大数据平台，主要是实现海量非结构化的客户行为数据的处理和分析，主要数据类型包括如下。

- 客户交易日志信息，主要来自银行信用卡系统、综合业务系统和网上银行、移动银行、电话银行等。
- 客户投诉信息，来自呼叫中心，以语音数据为主。
- 客户互动信息，来自企业微博、企业博客，包括第三方合作的社交媒体，如腾讯微信。

企业的 Hadoop 大数据平台物理上部署在企业级操作系统和虚拟化环境之上，主要部署的系统如下。

- 部署 HDFS 作为分布式文件系统，用于文件级操作。
- 部署 HBase 用于分布式的数据存储，采取基于<key,value>的列式存储。
- 采用 Hadoop MapReduce 做非结构数据的批量处理。
- Hive 与 Impala（正试用）的整合构成非结构化数据查询和分析的基础。
- 采用原有的数据分析软件 SAS 做数据分析和可视化，在此基础上加强文本分析和流分析工具。
- 采用 Hadoop 的 Eclipse-Plugin 和 IBM 的 Eclipse 作为集成化的编译环境。

逻辑上，Hadoop 大数据平台可以直接从业务系统、管理系统、外部系统采集非结构化数据，可以与企业级数据仓库实现连接，以实现商业智能分析。如图 3-73 所示为银行大数据应用平台的总体架构。



数据来源：CIOManage（赛智时代）

图 3-73　银行大数据应用平台的总体架构

3.5.2　技术环境安装与配置

Hadoop 技术环境安装与配置信息参见：

- Hadoop 快速入门[51]对初次使用者。
- Hadoop 集群搭建[52]对大规模分布式集群。

作为企业级应用，可以选用 Hadoop 发行版（Hadoop Distribution）。发行版就是基于开源的 Apache Hadoop 进行改造的商业解决方案，其中包括一系列定制的管理工具和软件。用户可以根据自己的需求选择不同的 Hadoop 供应商。

Cloudera 发行版。Cloudera 的 Hadoop 管理工具非常成熟，提供了应用导航功能，在 Hadoop 领域有非常多的应用案例，对于 Hadoop 的发展起着重要作用，拥有广泛的合作伙伴支持，专业的 Hadoop 产品提供商。Cloudera 发行版包括 Hadoop 1 和 Hadoop 2 两种选择，但 Cloudera 不建议将 Hadoop 2 投入生产环境。

Hortonworks 发行版。Hortonworks 拥有大量的 Hadoop 专家，同时对 Hadoop 的发展也起到了重要作用，也拥有广泛的合作伙伴支持，专门的 Hadoop 提供商。对专有代码的依赖低于 Cloudera，因此用户不必担心“厂商锁定”问题。Hortonworks 的 Hadoop 1（不包括 YARN、HCatalog 等），技术比较成熟且能够投入生产环境。

Intel 发行版。Intel 的 Hadoop 性能很好，Hadoop 发行版最先进入中国市场，有雄厚的研究力量保证本地化服务。

EMC Greenplum HD。使用 Greenplum 数据库的用户可以直接选择 Greenplum HD。

MapR 发行版。从某种程度上说，MapR 的 Hadoop 发行版在性能方面具备优势。

IBM 发行版。信任 IBM 的服务，可以选择它的 Hadoop 发行版。

在本案例中，银行一直在选用 IBM 的解决方案，在商业化应用方面把 IBM 的大数据解决方案作为选择之一。

作为技术人员，可以先行学习和适用 Hadoop。学习阶段可以采用 Apache 的开源的

[51] http://cn.hadoop.org/doc/quickstart.html。

[52] http://cn.hadoop.org/doc/cluster_setup.html。

Hadoop，安装方法[53、54]如下。

通常，集群里的一台机器被指定为 NameNode，另一台不同的机器被指定为 JobTracker。这些机器是 Master。集群中其他的机器是 DataNode，也作为 TaskTracker。这些机器是 Slave。

1. 先决条件

确保在集群中的每个节点上都安装了所有必需的软件：Sun-JDK 、ssh、Hadoop。Hadoop 是基于 Java，并且是通过 ssh 来与各个机器通信的。

（1）安装正式发行的 Java 版本。

（2）ssh 必须安装且保证 sshd 一直运行，以便用 Hadoop 脚本管理远端 Hadoop 守护进程。

2. 实验环境搭建

（1）准备工作

操作系统：Ubuntu。Ubuntu 这个操作系统很简洁，容易配置，而且占用硬盘空间比较小，适合实验环境。选用默认配置。

虚拟机环境部署：Vmvare。选择稳定的虚拟机环境。例如 VMware-workstation-6.5.1-126130，官网上下载[55]。

在 VMvare 安装好一台 Ubuntu 虚拟机后，可以导出或者复制出另外两台虚拟机。

安装注意事项：保证虚拟机的 IP 和主机的 IP 在同一个 IP 段，这样几个虚拟机和主机之间可以相互通信。为了保证虚拟机的 IP 和主机的 IP 在同一个 IP 段，虚拟机连接设置为桥连。

准备机器：一台 Master，若干台 Slave，配置每台机器的/etc/hosts 保证各台机器之间通过机器名可以互访，例如：

---

[53] http://hadoop.apache.org/common/docs/r0.19.2/cn/cluster_setup.html。

[54] http://blog.csdn.net/hguisu/article/details/7237395。

[55] VMVARE 官方下载地址：http://download.csdn.net/download/livit/988102。

10.64.56.76 node1（master）
10.64.56.77 node2 （slave1）
10.64.56.78 node3 （slave2）

如表 3-25 所示为对应示例的主机信息。

表 3-25　对应示例的主机信息

机　器　名	IP 地址	作　　用
Node1	10.64.56.76	NameNode、JobTracker
Node2	10.64.56.77	DataNode、TaskTracker
Node3	10.64.56.78	DataNode、TaskTracker

为保证环境一致要先安装好 JDK 和 ssh。

（2）安装 JDK

```
$ sudo apt-get install sun-java6-jdk1.2.3
```

这个安装，Java 执行文件自动添加到/usr/bin/目录。

验证 shell 命令：

```
java -version
```

看是否与版本号一致。

（3）下载、创建用户

```
$ useradd hadoop
$ cd /home/hadoop
```

在所有的机器上都建立相同的目录，也可以建立相同的用户，最好是以该用户的 home 路径来做 hadoop 的安装路径。

例如，在所有的机器上的安装路径都是：/home/hadoop/hadoop-0.20.203，这个不需要 mkdir，在/home/hadoop/下解压 hadoop 包的时候，会自动生成。（当然可以安装/usr/local/目录下，例如/usr/local/hadoop-0.20.203/，chown -R hadoop /usr/local/hadoop-0.20.203/，chgrp -R hadoop /usr/local/hadoop-0.20.203/。最好不要使用 root 安装，因为不推荐各个机器之间使用 root 访问。）

（4）安装 ssh 和配置

第 1 步，安装 ssh

```
$sudo apt-get install ssh
```

这个安装完后，可以直接使用 ssh 命令了。

```
$ netstat  -nat
```

查看 22 号端口是否开启了。

执行 ssh 命令：

```
ssh localhost
```

输入当前用户的密码，回车就可以。说明安装成功，同时 ssh 登录需要密码。（这种默认安装方式完成后，默认配置文件是在/etc/ssh/目录下。sshd 配置文件是/etc/ssh/sshd_config）。

注意事项：在所有机器上都需要安装 ssh。

第 2 步，配置 ssh。

在 Hadoop 启动以后，NameNode 是通过 ssh（Secure Shell）来启动和停止各个 DataNode 上的各种守护进程的，这就需要在节点之间执行指令的时候不输入密码，故需要配置 ssh 运用无密码公钥认证的形式。

以当前配置的 3 台机器为例，node1 是主节点，需要连接 node2 和 node3。要确定每台机器上都安装了 ssh，并且 DataNode 机器上 sshd 服务已经启动。

说明：[hadoop@hadoop～]$ssh-keygen -t rsa

这个命令将为 hadoop 上的用户 hadoop 生成其密钥对，询问其保存路径时直接回车键采用默认路径，当提示要为生成的密钥输入 passphrase 的时候，直接回车，也就是将其设定为空密码。生成的密钥对 id_rsa、id_rsa.pub 默认存储在/home/hadoop/.ssh 目录下，然后将 id_rsa.pub 的内容复制到每个机器（也包括本机）的/home/dbrg/.ssh/authorized_keys 文件中，如果机器上已经有 authorized_keys 这个文件了，就在文件末尾加上 id_rsa.pub 中的内容，如果没有 authorized_keys 这个文件，直接复制过去就行。

第 3 步，首先设置 NameNode 的 ssh 为无须密码自动登录的。

切换到 hadoop 用户（保证用户 hadoop 可以无须密码登录，因为我们后面安装的 hadoop 属主是 hadoop 用户。）

```
$ su hadoop
$ cd /home/hadoop
$ ssh-keygen -t rsa
```

然后一直按回车键。

完成后，在 home 根目录下会产生隐藏文件夹.ssh：

```
$ cd .ssh
```

之后用 ls 命令查看文件：

```
$ cp id_rsa.pub authorized_keys
```

测试：

```
$ssh localhost
```

或者：

```
$ ssh node1
```

第 1 次 ssh 会有提示信息：

```
The authenticity of host 'node1 (10.64.56.76)' can't be established.
RSA key fingerprint is 03:e0:30:cb:6e:13:a8:70:c9:7e:cf:ff:33:2a:67:30.
Are you sure you want to continue connecting (yes/no)?
```

输入“yes”来继续。这会把该服务器添加到你的已知主机的列表中。这步之后，发现链接成功，并且无须密码。

第 4 步，复制 authorized_keys 到 node2 和 node3 上。

为保证 node1 可以无须密码自动登录到 node2 和 node3，先在 node2 和 node3 上执行：

```
$ su hadoop
$ cd /home/hadoop
$ ssh-keygen -t rsa
```

一直按回车键。

然后回到 node1，复制 authorized_keys 到 node2 和 node3：

```
[hadoop@hadoop .ssh]$ scp authorized_keys node2:/home/hadoop/.ssh/
[hadoop@hadoop .ssh]$ scp authorized_keys node3:/home/hadoop/.ssh/
```

这里会提示输入密码，输入 hadoop 账号密码就可以了。

改动你的 authorized_keys 文件的许可权限：

```
[hadoop@hadoop .ssh]$chmod 644 authorized_keys
```

测试：

```
$ ssh node2
```

或：

```
$ ssh node3
```

（第 1 次需要输入“yes”）。

如果不需要输入密码则配置成功，如果还需要输入密码，请检查上面的配置是不是正确。

（5）安装 Hadoop

切换为 hadoop 用户：

```
$ su hadoop
wgethttp://apache.mirrors.tds.net//hadoop/common/hadoop-0.20.203.0/hadoop-0.20.203.0rc1.tar.gz
```

下载安装包后，直接解压缩安装即可：

```
$ tar -zxvf hadoop-0.20.203.0rc1.tar.gz
```

注意：

①安装 Hadoop 集群通常要将安装软件解压缩到集群内的所有机器上，并且安装路径要一致，如果我们用 HADOOP_HOME 指代安装的根路径，通常，集群里的所有机器的 HADOOP_HOME 路径相同。

②如果集群内机器的环境完全一样，可以在一台机器上配置好，然后把配置好的软件，如 hadoop-0.20.203，整个文件夹复制到其他机器的相同位置。

③可以将 Master 上的 Hadoop 通过 scp 复制到每一个 Slave 相同的目录下，同时根据每一个 Slave 的 Java_HOME 的不同修改其 hadoop-env.sh。

④为了方便，使用 hadoop 命令或者 start-all.sh 等命令，修改 Master 上的/etc/profile，新增以下内容：

```
export HADOOP_HOME=/home/hadoop/hadoop-0.20.203
    export PATH=$PATH:$HADOOP_HOME/bin
```

修改完毕后，执行 source /etc/profile 来使其生效。

（6）配置 conf/hadoop-env.sh 文件

```
export JAVA_HOME=/usr/lib/jvm/java-6-sun/
```

这里修改为你的 JDK 的安装位置。

测试 Hadoop 安装：

```
Bin/hadoop jar hadoop-0.20.2-examples.jarwordcount conf/  /tmp/out
```

3. 集群配置（所有节点相同）

（1）配置文件：conf/core-site.xml

```
<?xml version="1.0"?>
< ?xml-stylesheet type="text/xsl"href="configuration.xsl"?>
< configuration>
< property>
 <name>fs.default.name</name>
  <value>hdfs://node1:49000</value>
< /property>
< property>
  <name>hadoop.tmp.dir</name>
 <value>/home/hadoop/hadoop_home/var</value>
< /property>
< /configuration>
```

fs.default.name 是 NameNode 的 URI，hdfs://主机名:端口/。

hadoop.tmp.dir：Hadoop 的默认临时路径，这个最好配置，如果在新增节点或者其他情况下莫名其妙地 DataNode 就启动不了，就删除此文件中的 tmp 目录即可。不过如果删除了 NameNode 机器的此目录，那么就需要重新执行 NameNode 格式化的命令。

（2）配置文件：conf/mapred-site.xml

```
<?xmlversion="1.0"?>
<?xml-stylesheettype="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>node1:49001</value>
</property>
<property>
<name>mapred.local.dir</name>
<value>/home/hadoop/hadoop_home/var</value>
</property>
</configuration>
```

mapred.job.tracker 是 JobTracker 的主机（或者 IP）和端口，主机：端口值设为 nodel:49001。

（3）配置文件：conf/hdfs-site.xml

```
<?xmlversion="1.0"?>
<?xml-stylesheettype="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
<name>dfs.name.dir</name>
<value>/home/hadoop/name1, /home/hadoop/name2</value> #hadoop 的 name 目录路径
<description>  </description>
</property>
<property>
<name>dfs.data.dir</name>
<value>/home/hadoop/data1, /home/hadoop/data2</value>
<description> </description>
</property>
<property>
  <name>dfs.replication</name>
  <!-- 我们的集群又两个节点，所以 rep 两份 -->
  <value>2</vaue>
</property>
</configuration>
```

①dfs.name.dir 是 NameNode 持久存储名字空间及事务日志的本地文件系统路径。当这个值是一个逗号分割的目录列表时，nametable 数据将会被复制到所有目录中做冗余备份。

②dfs.data.dir 是 DataNode 存放块数据的本地文件系统路径，逗号分割的列表。当这

个值是逗号分割的目录列表时，数据将被存储在所有目录下，通常分布在不同设备上。

③dfs.replication 是数据需要备份的数量，默认是 3，如果此数大于集群的机器数会出错。

注意：此处的 name1、name2、data1、data2 目录不能预先创建，Hadoop 格式化时会自动创建，如果预先创建反而会有问题。

④配置 Master 和 Slave 主从节点

配置 conf/masters 和 conf/slaves 来设置主从节点，注意，最好使用主机名，并且保证机器之间通过主机名可以互相访问，每个主机名一行。

```
vi masters:
输入 node1
vi slaves:
分别输入 node2，node3
```

配置结束，把配置好的 hadoop 文件夹复制到其他集群的机器中，并且保证上面的配置对于其他机器而言正确，例如，如果其他机器的 Java 安装路径不一样，要修改 conf/hadoop-env.sh：

```
$ scp -r /home/hadoop/hadoop-0.20.203 root@node2: /home/hadoop/
```

4. Hadoop 启动

（1）格式化一个新的分布式文件系统

先格式化一个新的分布式文件系统：

```
$ cd hadoop-0.20.203
$ bin/hadoop namenode -format
```

成功情况下系统输出：

```
12/02/06 00:46:50 INFO namenode.NameNode:STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = ubuntu/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 0.20.203.0
STARTUP_MSG:     build   =http://svn.apache.org/repos/asf/hadoop/common/
```

```
branches/branch-0.20-security-203-r 1099333; compiled by 'oom' on Wed May 4
07:57:50 PDT 2011
    ************************************************************/
    12/02/0600:46:50 INFO namenode.FSNamesystem: fsOwner=root,root
    12/02/06 00:46:50 INFO namenode.FSNamesystem:supergroup=supergroup
    12/02/06 00:46:50 INFO namenode.FSNamesystem:isPermissionEnabled=true
    12/02/06 00:46:50 INFO common.Storage: Imagefile of size 94 saved in 0 seconds.
    12/02/06 00:46:50 INFO common.Storage: Storagedirectory /opt/hadoop/hadoopfs/name1
has been successfully formatted.
    12/02/06 00:46:50 INFO common.Storage: Imagefile of size 94 saved in 0 seconds.
    12/02/06 00:46:50 INFO common.Storage: Storagedirectory /opt/hadoop/hadoopfs/
name2 has been successfully formatted.
    12/02/06 00:46:50 INFO namenode.NameNode:SHUTDOWN_MSG:
    /************************************************************
    SHUTDOWN_MSG: Shutting down NameNode atv-jiwan-ubuntu-0/127.0.0.1
    ************************************************************/
```

查看输出，保证分布式文件系统格式化成功。

执行完后可以到 Master 机器上看到/home/hadoop//name1 和/home/hadoop//name2 两个目录。在主节点 Master 上面启动 Hadoop，主节点会启动所有从节点的 Hadoop。

（2）启动所有节点

◎ 启动方式 1（同时启动 HDFS 和 MapReduce）：

```
    $ bin/start-all.sh
```

系统输出：

```
    starting namenode, logging to
     /usr/local/hadoop/logs/hadoop-hadoop-namenode-ubuntu.out
    node2: starting datanode, loggingto
    /usr/local/hadoop/logs/hadoop-hadoop-datanode-ubuntu.out
    node3: starting datanode, loggingto
     /usr/local/hadoop/logs/hadoop-hadoop-datanode-ubuntu.out
    node1: starting secondarynamenode,logging to
     /usr/local/hadoop/logs/hadoop-hadoop-secondarynamenode-ubuntu.out
    starting jobtracker, logging to/usr/local/hadoop/logs/hadoop-hadoop-jobtracker-
ubuntu.out
    node2: starting tasktracker,logging to
    /usr/local/hadoop/logs/hadoop-hadoop-tasktracker-ubuntu.out
    node3: starting tasktracker,logging to
```

```
     /usr/local/hadoop/logs/hadoop-hadoop-tasktracker-ubuntu.out
    As you can see in slave's output above, it will automatically format it's
storage directory(specified by dfs.data.dir) if it is not formattedalready. It
will also create the directory if it does not exist yet.
```

◎　启动方式 2：

启动 Hadoop 集群需要启动 HDFS 集群和 MapReduce 集群。

在分配的 NameNode 上，运行下面的命令启动 HDFS（单独启动 HDFS 集群）：

```
$ bin/start-dfs.sh
```

bin/start-dfs.sh 脚本会参照 NameNode 上${HADOOP_CONF_DIR}/slaves 文件的内容，在所有列出的 Slave 上启动 DataNode 守护进程。

在分配的 JobTracker 上，运行下面的命令启动 MapReduce（单独启动 MapReduce）：

```
$bin/start-mapred.sh
```

bin/start-mapred.sh 脚本会参照 JobTracker 上${HADOOP_CONF_DIR}/slaves 文件的内容，在所有列出的 Slave 上启动 TaskTracker 守护进程。

（3）关闭所有节点

从主节点 Master 关闭 Hadoop，主节点会关闭所有从节点的 Hadoop。

```
$ bin/stop-all.sh
```

Hadoop 守护进程的日志写入到 ${HADOOP_LOG_DIR}目录（默认是 ${HADOOP_HOME}/logs）。

${HADOOP_HOME}就是安装路径。

5. 测试

（1）浏览 NameNode 和 JobTracker 的网络接口，它们的地址默认为：

```
NameNode - http://node1:50070/
JobTracker - http://node2:50030/
```

（2）使用 netstat -nat 查看端口 49000 和 49001 是否正在使用。

（3）使用 jps 查看进程。要想检查守护进程是否正在运行，可以使用 jps 命令（这是

用于 JVM 进程的 ps 实用程序）。这个命令列出 5 个守护进程及其进程标识符。

（4）将输入文件复制到分布式文件系统：

```
$ bin/hadoop fs -mkdir input
$ bin/hadoop fs -put conf/core-site.xml input
```

运行发行版提供的示例程序：

```
$ bin/hadoop jar hadoop-0.20.2-examples.jar grep input output 'dfs[a-z.]+'
```

其中：bin/hadoop jar（使用 hadoop 命令运行 jar 包），hadoop-0.20.2_examples.jar（jar 包的名字），grep input output 'dfs[a-z.]+'是运行 hadoop 示例程序中的 grep，对应的 hdfs 上的输入目录为 input、输出目录为 output。Grep 是一个 MapReduce 程序，它计算输入中的一个 Regex 的匹配数。

将输出文件从分布式文件系统复制到本地文件系统查看：

```
$ bin/hadoop fs -get output output
$ cat output/*
```

或者在分布式文件系统上查看输出文件：

```
$ bin/hadoop fs -cat output/*
```

统计结果：

```
root@v-jiwan-ubuntu-0:~/hadoop/hadoop-0.20.2-bak/hadoop-0.20.2#bin/hadoop
fs -cat output/part-00000
3       dfs.class
2       dfs.period
1       dfs.file
1      dfs.replication
1       dfs.servers
1       dfsadmin
```

6. HDFS 常用操作

hadoopd fs -ls 列出 HDFS 下的文件。

hadoop dfs -ls in 列出 HDFS 下某个文档中的文件。

hadoop dfs -put test1.txt test 上传文件到指定目录并且重新命名，只有所有的 DataNode

都接收完数据才算成功。

hadoop dfs -get in getin　从 HDFS 获取文件并且重新命名为 getin，同 put 一样可操作文件也可操作目录。

hadoop dfs -rmr out　从 HDFS 上删除指定文件。

hadoop dfs -cat in/*　查看 HDFS 上 in 目录的内容。

hadoop dfsadmin -report　查看 HDFS 的基本统计信息，结果如下。

- hadoop dfsadmin -safemode leave　退出安全模式。
- hadoop dfsadmin -safemode enter　进入安全模式。

7. 添加节点

可扩展性是 HDFS 的一个重要特性，首先在新加的节点上安装 Hadoop，然后修改 $HADOOP_HOME/conf/master 文件，加入 NameNode 主机名，然后在 NameNode 节点上修改$HADOOP_HOME/conf/slaves 文件，加入新加节点主机名，再建立到新加节点无密码的 ssh 连接。

运行启动命令：

```
start-all.sh
```

然后可以通过（node 的主机名）:50070 查看新添加的 DataNode

8. 负载均衡

在 DataNode 节点上选择策略重新平衡 DataNode 上的数据块的分布，运行：

```
Start -balancer.sh
```

第 4 章

大数据与企业级应用的整合策略

本章阐述大数据技术与现有企业级应用的整合策略。

企业级 IT 应用在近十年的快速发展中经历了 5 个重要里程碑。第 1 个是标准化，即采用标准化的 IT 基础设施，包括标准的服务器、存储、网络设备、系统软件（操作系统、数据库、中间件软件）等，基于 X86 标准的服务器、Linux 操作系统被广泛应用。第 2 个是集成和整合，通过对物理基础设施的集成，例如推出刀片服务器、数据仓库一体机等，来满足成本、可用性、性能、可扩展性、互操作性和安全性（简称为 CAPSIMS）的要求。第 3 个是虚拟化，通过将服务器、存储等计算资源集中到资源池中，以虚拟化的服务提供，通过对资源均衡来优化各种业务服务的交付。第 4 个是自动化，实现自动维护，满足应用级服务级别协议 SLAs（要求系统能自配置、自愈等）。第 5 个是云计算，包括企业私有云和公共云部署。当前，企业级应用已经进入第 6 个里程碑——大数据应用，对海量非结构化数据基于 Hadoop 大数据平台进行处理和分析，并进行预测分析和可视化展现。大数据应用需要考虑与这些现有企业级应用系统的集成和整合。

4.1　大数据传输、整合和流程管理平台

4.1.1　数据传输

1. Sqoop

（1）概述

Apache Sqoop 项目旨在协助 RDBMS 与 Hadoop 之间进行高效的数据交流。用户可以在 Sqoop 的帮助下，轻松地把结构化的数据存储（包括结构化的数据库、数据仓库、基于文档的系统）的数据导入到 Hadoop 系统（以及与其相关的系统，如 HBase 和 Hive）中；同时也可以把数据从 Hadoop 系统里抽取并导出到结构化数据库里，如图 4-1 所示。所以，Sqoop 也能帮助用户在 RDBMS 和 HBase、Hive 之间实现数据交流。自从 2009 年 5 月作为 Hadoop 的补丁首次发布以来，Apache Sqoop 逐步迅速发展，并于 2012 年 3 月正式成为 Apache 的顶级项目。

（2）Sqoop 机理

Sqoop 使用一个基于连接器的体系架构，这种架构支持用于与外部系统实现连接的插件。Sqoop 能够用一种快速并行方式转移数十亿行的数据进入 Hadoop，它用 MapReduce 框架来并行传输数据。Sqoop 要么直接将数据放入由 Hadoop 分布式文件系统（HDFS）管理的数据存储空间，或者也可以应用到其他 Hadoop 应用程序中，如 HBase 或 Hive。被转移的数据集被切分成不同的分区（partition），一个只有 Map 的作业被启动，每个独立的 Mapper 负责转换数据集的每个切分。数据的每个记录被以一种类型安全的方式被处理，因为 Sqoop 用数据库的元数据来推导数据类型。

（3）Sqoop 支持的数据库

Sqoop 提供一个可扩展的机制来用整合高性能的外部系统连接器。理论上，Sqoop 支持任何一款支持 JDBC 规范的数据库和数据仓库，如 Oracle、DB2、SQL Server、MySQL、PostgreSQL、Netezza、Teradata、Microsoft PDW 等。

在使用 Sqoop 连接关系型数据库前，首先需要把相关的 JDBC 驱动复制到$SQOOP_HOME/lib 文件夹下，然后在 connect 参数后指定好数据库连接的 URL，如“-connect jdbc:mysql://localhost/SAMPLE”。

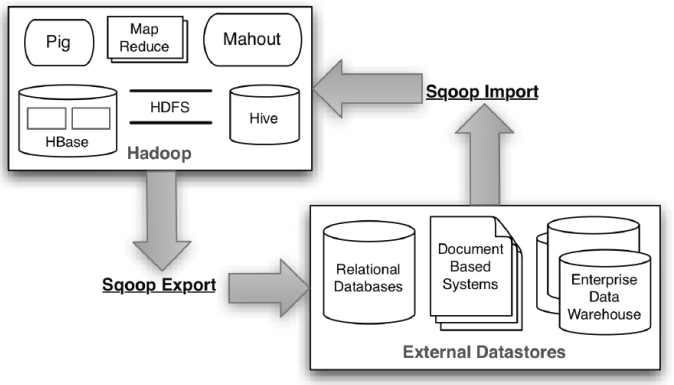


图 4-1　Sqoop 的作用

（4）Sqoop 支持的文件类型

Sqoop 能够将数据库的数据导入到 HDFS 上，并保存为多种文件类型。常见的有分隔符文本类型、Avro 二进制类型以及序列化文件类型等。

（5）Sqoop 环境准备

将 Sqoop 项目从 Apache Sqoop 官网[1]上下载到本地，然后直接解压缩到本地系统。Sqoop 运行于 Hadoop，因此首先需要确保 Hadoop 环境已经安装好，并且将 Hadoop 的安装路径（HADOOP_HOME）添加到系统环境变量中，让 Sqoop 能够找到它。

（6）Sqoop 运行

Sqoop 架构非常简单，其整合了 Hive、HBase 和 Oozie，通过 MapReduce 任务来传输数据，从而提供并发特性和容错性，如图 4-2 所示。

Sqoop 的流程包括：预处理、数据转换和后处理 3 个阶段。预处理阶段包括连接器的选择、代码的生成和作业的配置。数据转换阶段包括作业的提交和处理。后处理阶段包括数据的清洗和 Hive 命令的执行。

◎　预处理阶段

预处理阶段确定 JDBC/ODBC 等连接器类型后，进行元数据查询，生成 SQL 类型的表，并把 SQL 类型转换为 Java 代码，生成 SqoopRecord，如图 4-3 所示。随后配置 MapReduce 作业。

[1] http://sqoop.apache.org/。

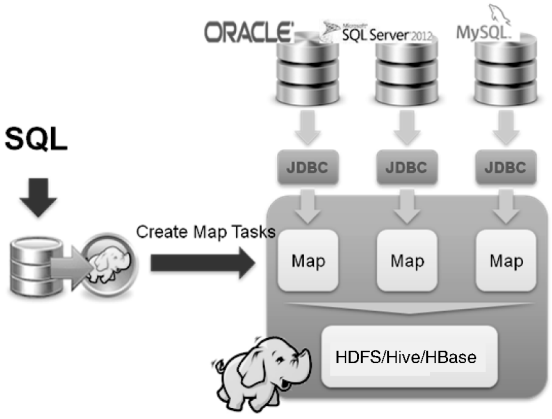


图 4-2　Sqoop 原理

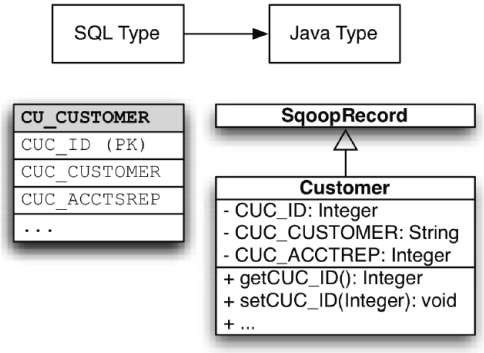


图 4-3　Sqoop 预处理阶段 —— 代码生成

例如，运行如下命令：

```
$ sqoop import --connect jdbc:mysql://localhost/acmedb \
--table ORDERS --username test --password ****
```

能把 ORDERS 表输入到 HDFS 的目录中，自动生成 ORDERS.java 便于使用：

```
$ sqoop export --connect jdbc:mysql://localhost/acmedb \
--table ORDERS_CLEAN --username test --password **** \
--export-dir /user/arvind/ORDERS
```

能将 HDFS 目录输出到表 ORDERS。

Sqoop 能自动生成数据类型，基本映射表如表 4-1 所示。

表 4-1　SQL 类型和 Java 类型的映射

SQL 类型	JAVA 类型
INTEGER	Java.lang.Integer
VARCHAR	Java.lang.String
LONGVARCHAR	Java.lang.String
NCHAR	Java.lang.String
NUMERIC	Java.math.BigDecimal
DECIMAL	Java.math.BigDecimal
BOOLEN	Java.lang.Boolean
DATE	Java.sql.Date
……	……

Sqoop 能从表中读取信息并生成 Java 类，这些类能够用于随后的 MapReduce 处理阶段。

Sqoop 能选择不同的表（或者列的子集），能读数据库中所有的表，并支持大多数 JDBC 标准类型和空值。

◎　数据转换阶段

主要是提交作业，Mapper 做并行的数据转换，如图 4-4 所示。

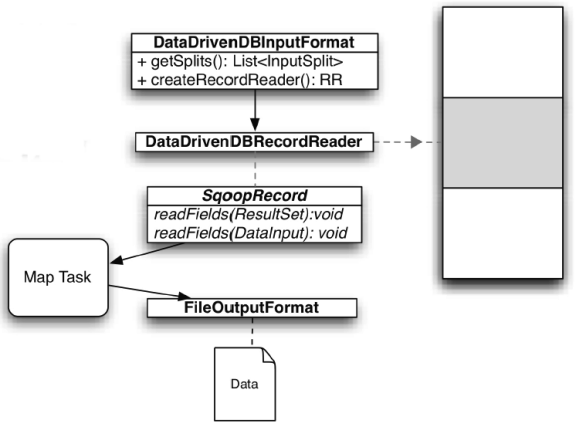


图 4-4　Sqoop 数据转换

DBInputFormat 函数能连接到 JDBC 接口，能提供接口来使用任意的输入查询、表和

数据库，也能选择表外的记录或任意的查询。记录被写到 DBWritable，作为给 Mapper 的值。DBWritable 定义一个类来拥有数据库中一行，必须能从 JDBC ResultSet 读数据到域。必须能写 JDBC PreparedStatement，应该也可实施常规的 Writable。

◎　数据后处理阶段

主要是数据清洗，并执行 Hive 命令。

与 Hive 集成，生成 CREATE TABLE / LOAD DATA INPATH 脚本。数据被输入到 HDFS 中，自动执行 Hive 脚本；随后的步骤是装载数据到 partition，如图 4-5 所示。

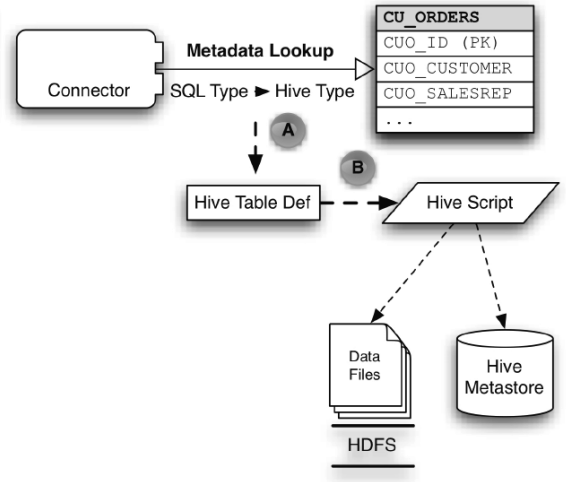


图 4-5　Sqoop 数据后处理

例如，如下语句把表 ORDERS 输入 Hive：

```
$ sqoop import --connect jdbc:mysql://localhost/acmedb \
--table ORDERS --username test --password **** --hive-import
```

一旦输入完成，用户就能看到和操作 ORDERS 表，就像 Hive 中的其他表一样。

Sqoop 也可以实现结构化数据库导入到 HBase 表中，所有输入到 HBase 的数据被转换成为 string 并作为 UTF-8 类型插入。例如，把表 ORDERS 输入 HBase。

```
$ sqoop import --connect jdbc:mysql://localhost/acmedb \
--table ORDERS --username test --password **** \
--hbase-create-table --hbase-table ORDERS --column-family mysql
```

目前版本的 Sqoop 已经被很多 Hadoop 用户在生产任务中应用。网上营销 Coupons 公司用该软件在 Hadoop 和 IBM 的 Netezza 数据仓库设备之间进行数据交换，该企业可以查询它的结构化数据库和使用 Sqoop 将结果输入到 Hadoop 中。教育公司 Apollo 集团也用该软件，不仅从数据库中提取数据，而且 Hadoop 的工作结果返回关系型数据库中。

2. Flume

Apache Flume 帮助流数据源，如 Web 或应用服务器、网络服务或操作系统，直接装载实时数据到 HDFS、Hive 或 HBase 中。Flume 支持各种源协议，如 Avro、Thrift、Syslog 和 NetCat，也能处理任何分隔符文件格式。Flume 也具有在线数据转换和直接消息的能力。用 Flume，各种数据源能被收集并以任何格式存储在 Hadoop 中。

一个 Flume 事件被定义为一个数据流的基本数据单位，它携带日志数据（字节数组形式）并且携带有头信息。string 属性的集合。Flume 的最小独立运行单位称为 Agent，一个 Agent 就是一个 JVM 过程，它是由 Source（源）、Sink（接收地）和 Channel（通道）三大组件构成的，Agent 管理它的组件以完成事件流从一个外部源到下一个目的地（称为 hop）。

Flume Source 吸收来自外部源（如 Web 服务器）的事件。外部源发送数据到 Flume，采取目标 Flume Source 能够识别的格式。例如，一个 Avro Flume Source 能够用于接收 Avro 事件，它们来自 Avro 客户端或流中的其他从 Avro Sink 中发送事件的 Agent。类似的流也能被用 Thrift Flume Source 来定义，以接收从一个 Thrift Sink 或者 Flume Thrift RPC 客户端或者用 Flume Thrift 协议生成的任何语言所写的 Thrift 客户端。当一个 Flume Source 接收到一个事件，把它存储到一个或多个 Channel。Channel 是一个主动的存储，它保留事件直到被本地的文件系统退回。Sink 从 Channel 中移走事件，并把它放到一个外部的库，如 HDFS（通过 Flume HDFS Sink）或者把它推送到下一个 Flume Agent 的 Flume Source。给定 Agent 中的 Source 和 Sink 可以异步运行在 Channel 中的事件，如图 4-6 所示。

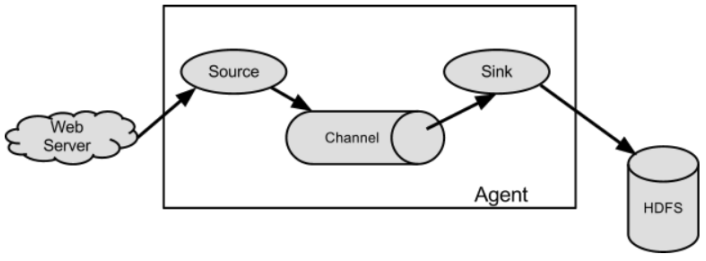


图 4-6　Flume 数据流

Flume 允许用户建立多个 hop 的流动，其中事件在到达目的地前在多个 Agent 之间流动，并最终能合并到一个 hop。它也允许 fan-in 和 fan-out 流动，实现多路传输，如图 4-7 所示，支持上下文路由和对失败 hop 的备份路由（失效）等。

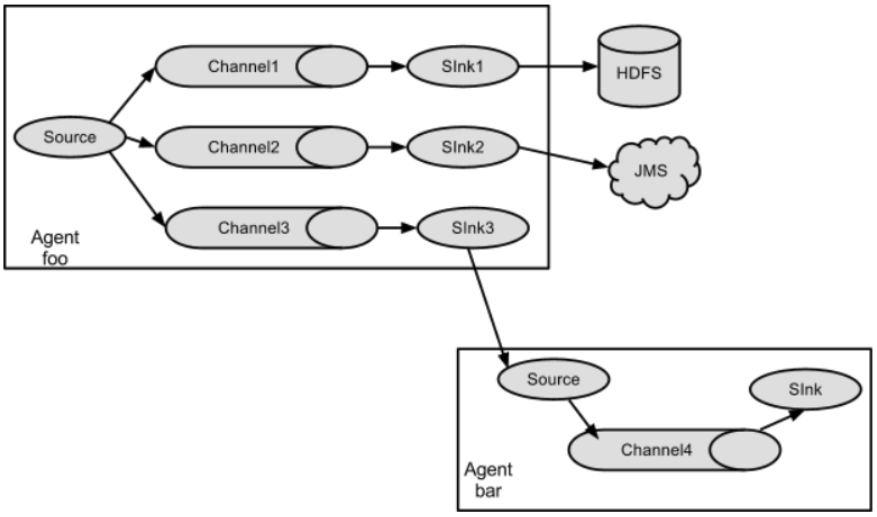


图 4-7　Flume 中的多路传输

篇幅所限，更多内容请读者参考 Flume 的用户指南[2]。

3. HttpFS

Apache Hadoop HttpFS 是一个服务，它能帮助 HTPP 访问到 HDFS，这是为了基于 REST（Representational State Transfer，简称 REST，表述性状态转移）的文件流动。HttpFS 使得集成 Hadoop 到一个面向服务架构中变得很简单。由于 HttpFS 支持与整个 Hadoop 堆栈（stack）和在线加密的 SSL 的完全的安全集成，HttpFS 提供了一个完全安全的网关来保护敏感数据，如客户的财务记录。

4.1.2　数据整合

Hadoop 因为其批处理的特点，最早曾使用 ETL 进行数据集成。但是，如果用基于 ETL

[2] http://flume.apache.org/FlumeUserGuide.html。

解决方案来运行与维护复杂的 Hadoop 平台上的所有的基础设施，就需要更全面的数据集成工具，例如 Informatica、Talend（开源）、Syncsort、CloverETL（开源）等公司的解决方案。多年来，这些公司努力建立最佳组合的 ETL 解决方案，即数据整合解决方案， 如图 4-8 所示。例如，Informatica 公司数据整合解决方案中的工具 HParser，它是一种针对 Hadoop 而优化的数据转换环境，支持灵活高效地处理 Hadoop 里面的任何文件格式，为 Hadoop 开发人员提供了即开即用的解析功能，以便处理复杂而多样的数据源，包括日志、文档、二进制数据或层次式数据，以及众多行业标准格式（如银行业的 NACHA、支付业的 SWIFT、金融数据业的 FIX 和保险业的 ACORD）。正如数据库内处理技术加快了各种分析方法，Informatica 同样将解析代码添加到 Hadoop 里面，以便充分利用所有这些处理功能。

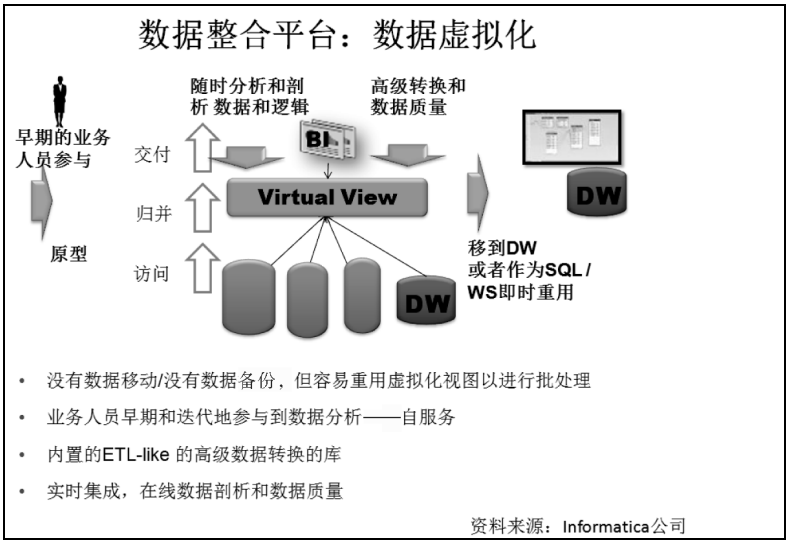


图 4-8　数据整合平台

多年来，数据整合所面临的一个挑战是不能跨组织和部门孤岛将企业级数据联系起来。这种整合对于大数据甚至更加重要，但也更加复杂。为了能够获得这个完整的视图，企业需要将关键数据从企业应用、云应用以及第三方数据提供商的系统中提取并集中起来。主数据管理（MDM）解决方案是为实现这个目标的解决方案，它能实现关键主题数据的整合，例如客户、产品、供应商、员工等，这些数据根据单一企业标准进行管理和共享。它从多个业务系统中整合最核心的、需要共享并保持一致的主数据，集中进行主数据的清洗和丰富，在整个企业范围内合并和维护唯一的、完整的和准确的主数据信息（客户、

供应商、员工、资产、设备、财务数据等），并以服务的方式把统一、完整、准确的主数据分发给企业范围内需要使用这些数据的业务系统、业务流程和决策支持系统，如图 4-9 所示。主数据管理是企业级大数据分析的一个基础性的工作。

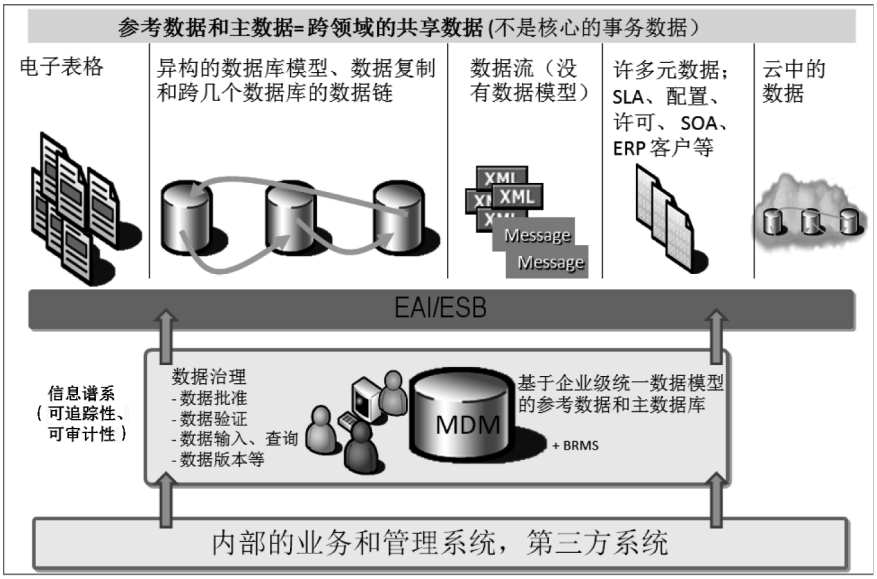


图 4-9　企业主数据管理模型

4.1.3　流程管理

Oozie 是 Hadoop 的工作流引擎。最早是在雅虎被设计出来，自 2011 年起成为 Apache 的孵化项目[3]。作为一个工作流引擎，如果有以下几种情形的任何一种情况，就可以采用 Oozie，包括有多个相互依赖的作业、需要按照规则来运行多个作业、需要检查数据的可用性、需要监控作业运行并进行服务支持等。

Oozie 工作流图是一组把动作连接起来的有向无环图（DAG）。Oozie 支持的动作包括：MapReduce 动作、Pig 动作、Java 动作、FS（HDFS）动作、E-mail 动作、Shell 动作、Hive 动作、Sqoop 动作、Sub-workflow 动作、写定制动作等，如图 4-10 所示。

[3] http://incubator.apache.org/oozie/。

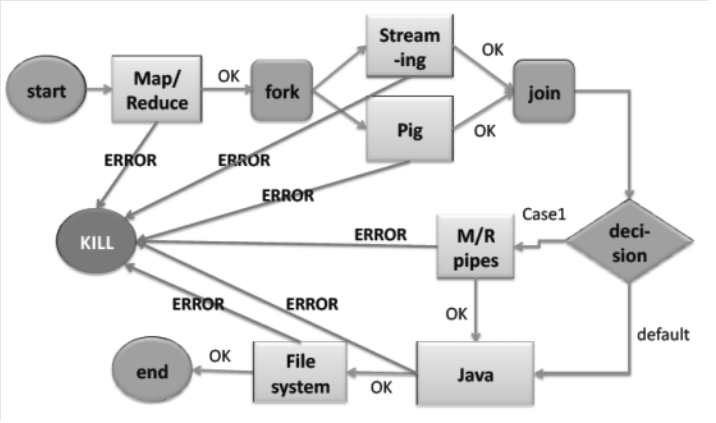


图 4-10　Oozie 工作流图示例

在 Oozie 中，动作的基本工作流控制就是：OK 和 ERROR，分别表示为<ok to="..."/>和<error to="..."/>，如下 XML 文件所示：

```
<workflow-app name="demo1" xmlns=url:oozie:workflow:0.3>
        <start to="act1"/>
        <action name="act1">
            …忽略….
            <ok to="act2"/>
            <error to="kill"/>
        </action>
        <kill name="kill">
            <message>custom message erro</message>
        </kill>
        <end name="end"/>
    <workflow/>
```

在启动<start…>、结束<end…>和消灭<kill…>等节点外，更多的工作流控制节点包括：<fork...>、<join ...>、<decision ...>，如图 4-11 所示为 Oozie 的控制节点。

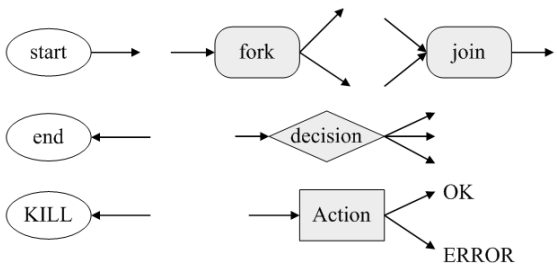


图 4-11　Oozie 的控制节点

下面是一段工作流配置 XML 文件：

```
<workflow-app name='my_wf'>
<start to='myprogram' />
<action name='myprogram'>
<map-reduce>
        <job-tracker>${jobTracker}</job-tracker>
    <name-node>${nameNode}</name-node>
        <streaming>
        <mapper>mymapper.pl</mapper>
        <reducer>/bin/wc</reducer>
        </streaming>
    <configuration>
        <property>
        <name>mapred.input.dir</name>
            <value>${hdfs_input_dir}</value>
        </property>
    </configuration>
</map-reduce>
    <ok to='yay' />
<error to='fail' />
</action>
<end name='yay' />
<kill name='fail'><message>failed</message></kill>
    </workflow>
```

这段 XML 就是执行如图 4-12 所示的工作流。

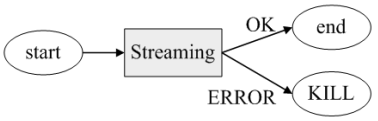


图 4-12　Oozie 工作流示例

协调器 Coordinator 是工作流中一个控制引擎，它在数据集可用时，按照规则执行一个工作流。如下的协调器配置文件，像一个 Crontab 脚本，执行每 60 分钟运行一次工作流的调度任务。

```
<coordinator-app name="demo2" frequency="60">
        start="2012-10-02T00:00Z" end="2012-10-04T24:00Z"…>
    <action>
        <workflow>
            <app-path>hdfs://xxxx:9000/myworkflow/demo1</app-path>
```

```
            </workflow>
        </action>
    </coordinator>
```

使用 Oozie 分为 3 个步骤。

第 1 步，在 HDFS 上配置你的工作流，包括：

- Oozie 作业定义（workflow.xml 文件）
- 工作流应用的代码：MapReduce/Pig 脚本/Streaming/Java 等
- 库（.so & .jar），如 streaming.jar、pig.jar、包括你自己的库
- 配置文件
- 所有其他必需的外部文件

第 2 步，提交作业，运行应用

```
$export OOZIE_URL=http://your.oozie.server:4080/oozie
$ oozie job -run -config job.properties
```

第 3 步，检查作业的状态

```
$ oozie job -info 0123-123456-oozie-wrkf-W
$ oozie job -log 0123-123456-oozie-wrkf-W
```

（用同样的方式来提交 Coordinator。）

Oozie 是基于服务器的工作流引擎，如图 4-13 所示。客户的命令发送给 Oozie 服务器，Oozie 服务器调动在 HDFS 中的工作流应用，所有文件在 HDFS。因此，Oozie 中动作总是远程的。Oozie 中，一个 Web 服务基于时间关系和数据关系启动作业。在工作流中没有循环。作业的恢复是被限制的，只支持部分作业再运行。

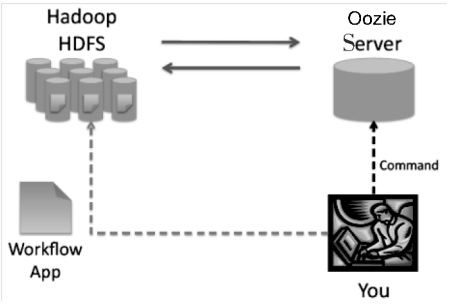


图 4-13　基于服务器的 Oozie

4.2　大数据与存储架构的整合

4.2.1　传统存储架构比较

传统存储架构有 DAS、SAN、NAS 和 iSCSI，在 Hadoop 集群上最自然的存储方式就是各个节点的本地 DAS，但大数据应用也需要更好地解决数据保护等问题，需要有效整合当前的存储架构，并对存储架构的发展提出了新的要求。图 4-14 和表 4-2 是对常用存储架构的比较。从存储架构比较结果看，FC-SAN 和 iSCSI 架构是提供块存储的外置设备，更适合关系型数据库的存储。NAS 是提供文件级共享的外置存储设备，则比较适合基于文件的非结构化数据的存储。

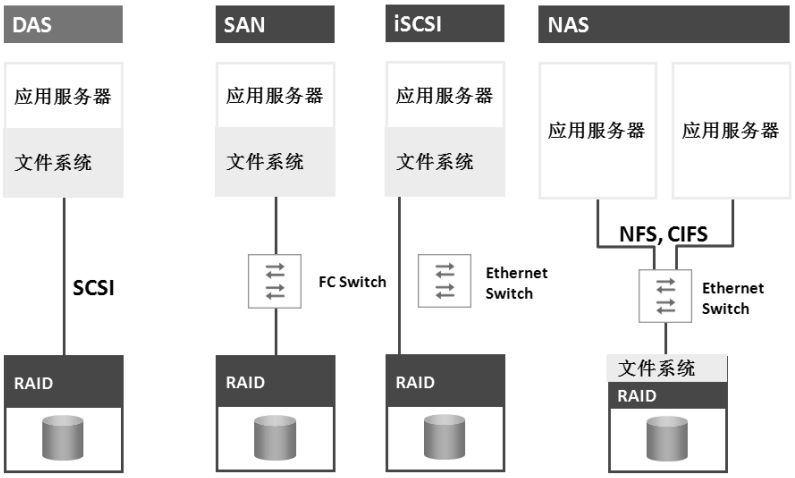


图 4-14　企业常用存储架构的比较

表 4-2　常用存储架构比较

	DAS	SAN	iSCSI	NAS
起源时间	20 世纪 80 年代	20 世纪 90 年代中后期	2001 年	20 世纪 90 年代
连接链路	无	光纤	以太网链路，TCP/IP 协议	以太网链路，TCP/IP 协议
存储网络	无	光纤+专用 FC 交换机	以太网+专用以太网交换机	以太网链路+专用以太网交换机
存储协议	SCSI	SCSI	SCSI	NFS、CIFS

续表

	DAS	SAN	iSCSI	NAS
存储空间	存储磁盘阵列本身难以扩展	多台前端服务器共用后端存储设备，后端存储空间以 LUN 形式提供给前端服务器，每台服务器拥有自己的存储空间	多台前端服务器共用后端存储设备，后端存储空间以 LUN 形式提供给前端服务器，每台服务器拥有自己的存储空间	多台前端服务器共享后端存储设备，后端 NAS 设备上的存储空间通过 CIFS、NFS 协议共享给前端主机
数据共享	不支持数据共享	不支持数据共享，每个 LUN 只能属于前端某一台服务器	不支持数据共享，每个 LUN 只能属于前端某一台服务器	可同时对同一目录或文件进行并发读写
存储空间控制	通过主机，需要消耗主机资源	通过主机，需要消耗主机资源	通过主机，需要消耗主机资源	通过磁盘阵列，不需要消耗主机资源
文件系统位置	直接连接存储设备	位于网络前端	位于网络前端	位于网络后端的存储系统
链路速率	无	2Gbps、4Gbps	1Gbps、10Gbps	1Gbps、10Gbps
存储容量	TB 级	TB 级	TB 级	TB 级
可扩展性	不好	一般	一般	较好，但有聚合设备（NAS 头）瓶颈
存储效率	高	高	中	中
可靠性	高	高	中	中
成本	低	高	中	中
管理复杂性	高	高	中	中
适用范围	主机模式	基于数据库的存储和访问	基于数据库的存储和访问	基于文件方式的存储和访问，这些文件需要被前端多套主机共享访问业务数据，大量数据需要共享和交换

4.2.2 大数据平台的存储架构的选择

大数据平台对存储架构的适合性分析如下。

本地 DAS。任何一个 Hadoop 平台所采用的存储本来就是本地 DAS 的模式。Hadoop

平台上的每个 DataNode 节点，计算和存储是在一起的，每个节点都有本地的 DAS。Hadoop 强调的是并行计算和大容量存储的能力。就 DAS 存储功能而言，在数据复制等方面，成本较高。因此，对于 Hadoop 计算中获得的高质量数据、重要文件和分析结果还是应该存储在 SAN 或者 NAS 中。

SAN。Hadoop 平台可以建立在 SAN 之上，对 HDFS 而言，SAN 看起来只是挂在 DataNode 上的一个局部的附加磁盘，但物理上它是 FC-SAN 中的磁盘阵列。采用 SAN 作为存储，增加了存储性能，但会牺牲一些 HDFS 文件系统本身的性能和功能。例如，HDFS 在 DataNode 之间的数据复制就显得多余了，磁盘阵列本身能够提供冗余。而且，HDFS 的扩展性受到限制，新扩展存储的成本也会更高。总之，如果企业有较富余的 SAN，SAN 是可以作为 Hadoop 的存储。

NAS。NAS 所依赖的网络文件系统（NFS/CIFS）本身就是一种分布式文件系统，只是与 HDFS 相比，网络文件系统更强调文件交换与共享。HDFS 则是一个高度耦合的、为 Hadoop 优化的、用于分布式计算的文件系统，它具有单一的全局命名空间，并行计算能力要求支持多个磁盘节点、对文件的多次访问。因此，就 Hadoop 平台而言，可以对 NameNode 采用 NAS 进行存储，但对 DataNode 采用传统的 NAS 显得必要性不足。

4.2.3　集群存储的发展

随着非结构化数据的海量增长，现有存储架构已经不能满足需求。例如，现有多台独立运行的部门级的 NAS 存储设备、SAN 存储设备，形成一个个“存储孤岛”，各部门间不能实现信息共享，存储系统的利用率低下。存储系统对业务支撑缺乏灵活度，不能进行灵活的扩展以满足快速变化的业务需求，确保数据可用及业务不中断。面对海量非结构化数据，现有存储系统缺乏大规模文件处理能力，导致出现性能瓶颈。而且，系统管理员需要管理多个存储系统。

在这种背景下，统一的、集群的、基于云的集群存储系统的需求被提出来，它要求：

- 可适应数据快速增长的存储可扩展性。
- 支持 PB 级大规模非结构化文档的存储和访问。
- 响应动态可变的业务需求。
- 加强数据共享与协作。

- 保护战略性信息资产。
- 高度自动化和简化的 IT 操作。

当前，集群存储是基于并行文件系统建立的新一代 NAS 集群系统。它通过 NAS 存储节点的扩展来实现集群存储，具有可扩展的海量存储空间（支持 PB 级存储）、大量文件存储和访问能力以及高速并行存取的存储效率。采用集群架构，存储系统的高可用性和容错性也得到了保证。集群存储的特点如下[4]。

1. 开放式架构（高扩展性）

它针对集群存储内部构成元素而言。一般集群存储应该包括存储节点、前端网络、后端网络等 3 个构成元素，每个元素都可以非常容易地采用业界最新技术而不用改变集群存储的架构，且扩展起来非常方便。

2. 分布式操作系统

所有对集群存储的操作都经由分布式操作系统统一调度和分发，分散到集群存储各个存储节点上完成。使用分布式操作系统带来的好处是各存储节点之间没有任何区别，没有主次、功能上的区别，所有存储节点功能完全一致，这样才能真正做到性能最优。

3. 统一命名空间

在集群存储中，统一命名空间强调的是同一个文件系统下的统一命名空间。它同样可以支持上 PB 级别的存储空间。

4. 易管理性

集群存储应该提供一种集中的、简便易用的管理方式，对客户端没有任何影响，采用业界标准的访问协议（比如 NFS、CIFS）访问集群存储。

集群存储的架构如图 4-15 所示。

[4] http://baike.baidu.com/view/4017608.htm。

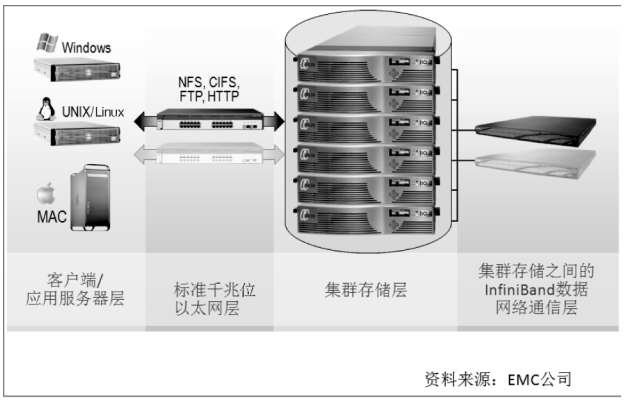


图 4-15　集群存储架构

5. 负载均衡

集群存储通过分布式操作系统的作用，会在前端和后端都实现负载均衡。前端访问集群存储的操作，通过几种负载均衡策略，将访问分散到集群存储的各个存储节点上。后端访问数据，通过开放式的架构和后端网络，数据会分布在所有节点上进行存放和读取。

6. 高性能

在高带宽、高并发访问的应用模式下，集群存储可以提供比传统存储架构更优的性能。但对于高 IOPS（每秒进行读写 I/O 操作的次数）、随机访问、小文件访问以及备份归档等其他类的应用，集群存储尚不能满足需求。

商业化的文件级集群存储系统有 EMC 公司的 Isilon、IBM 公司的 SONAS、NetApp 公司的 GX+ Data ONTAP、Panasass 等。

集群存储的优势主要体现在提高并行或分区 I/O 的整体性能，特别是工作流、读密集型以及大型文件的访问，通过采用更低成本的服务器来降低整体成本。从大数据应用需求看，集群存储设备具备为大数据存储和访问提供服务的能力。但不能因此牺牲 Hadoop 自身的性能。

4.2.4　基于 HDFS 的集群存储

集群存储系统既是一个分布式的文件系统，也是一个集群存储系统，相比分布式文件

系统，它增加了数据复制、数据备份恢复、数据消重等存储系统功能。因此，在当前企业应用环境下，整合 HDFS 和集群存储的优势，基于 HDFS 来建立集群存储，实现企业对大数据存储的支持，也是厂商推荐的一种选择。EMC 公司就推出了基于 Isilon 的 HDFS 系统，通过与 Hadoop 的整合来实现对大数据存储的支持。据 EMC 公司资料，它可以实现如下优点。

- 采用了集群存储，可以在某种程度保证 Hadoop 计算性能的情况下，实现存储和计算的分离。采用集群存储的 HDFS 后，Hadoop 可以直接用现有的数据。可以基于原有集群存储能力来实现计算能力的扩展。
- 集群存储本身有很多备份归档机制，可以对 Hadoop 作业用通过存储、快照、镜像等方式进行数据恢复。
- 基于存储集群的 HDFS，所有 NameNode 上的数据会分散地存放在所有的存储节点上，即使一个节点坏掉，数据仍然是非常安全的。并且所有的存储节点都可以通过域名解析出来，即使一个节点坏掉，其他的节点仍然可以提供 NameNode 的输入，NameNode 带来的单点故障会得到很好的解决。
- 集群存储本身支持 Hadoop 以外的计算能力需求，实现一份数据也可以同时提供各种协议的访问，可以带来很好的协议的支持。

当然，采用集群存储会牺牲 Hadoop 本身 HDFS 的优势，而且，可能带来在文件系统方面的重复的投资。在存储和计算能力的折中上，存储显得更为重要的时候，基于 HDFS 的集群存储是一个选择。如图 4-16 所示是 EMC 公司采用 GreenPlum HD 和 Isilon 集群构建 Hadoop 平台的一个示例。

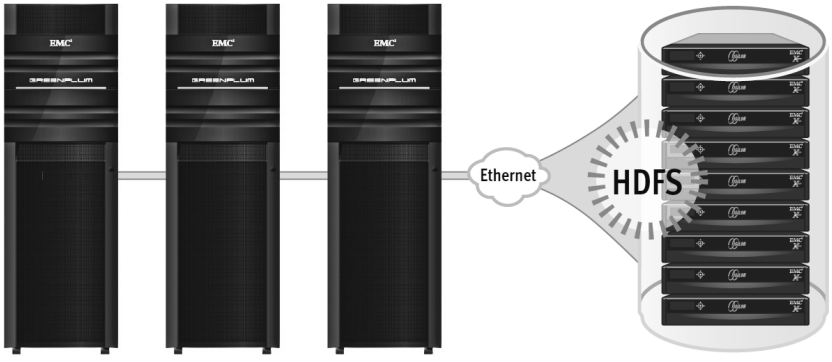


图 4-16　EMC 公司采用 GreenPlum 和 Isilon 集群构建 Hadoop 平台

4.2.5　固态硬盘（SSD）对内存计算的支持

固态硬盘可以为原始存储提供高性能、高吞吐率的支持。商业应用中数据库一体机或者大数据一体机均采用内存模块和闪存（SSD）来处理大数据。如果不关注成本，这是一个很好的解决方案。

总之，大数据平台也是整个数据中心架构的一个组成部分，需要与存储架构实现有效的集成，如图 4-17 所示（本图来源于 Oracle 公司）。

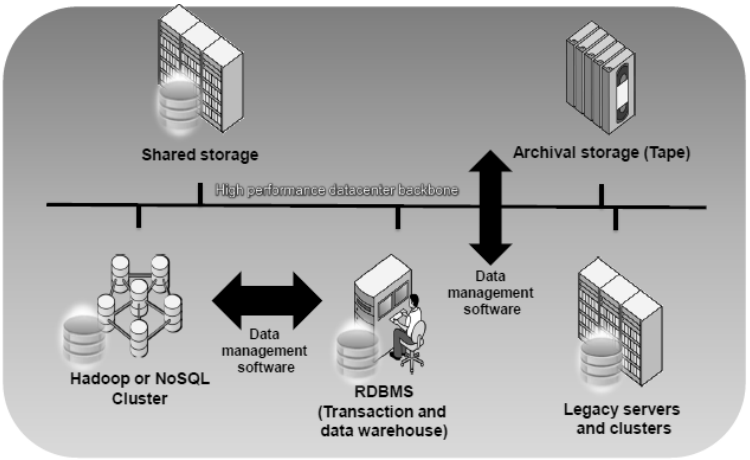


图 4-17　大数据与传统应用存储的整合

4.3　大数据与网络架构的发展

大数据技术平台是基于现有网络架构来实现分布式计算的。如图 4-18 所示是一个典型的云计算环境下的企业级网络架构。篇幅所限，这里不展开详细介绍。

但是，在大数据应用中，海量的各类大数据的快速传输（3V 特点）增加了网络中实时和大负载事务的数量。这对网络架构提出新的要求，支撑大数据应用的网络连接必须是健壮的，要足以保证数据能快速、高效地流动。第二，大数据是实时变化的，在一定时间内流向网络的数据量并不确定。因此，在大数据应用中，网络的流量模式可能是忽高忽低的脉冲式和可变的。当数据加载任务被请求时，数据必须通过网络来传输和分布到集群上。因此，必须有足够的网络资源池来支持大数据的网络传输和分布，否则，数据传输中的延

误就可能会很大。第三，灵活的交换机能力配置是获取网络效率的基本任务之一，通常大数据的网络配置可能需要 1GbE 的访问层交换能力。在近年来，成本性能比变得更加有效果，10GbE 服务器连接将更普遍，一些组织可能需要更新汇聚交换能力到 40GbE，甚至 100GbE。

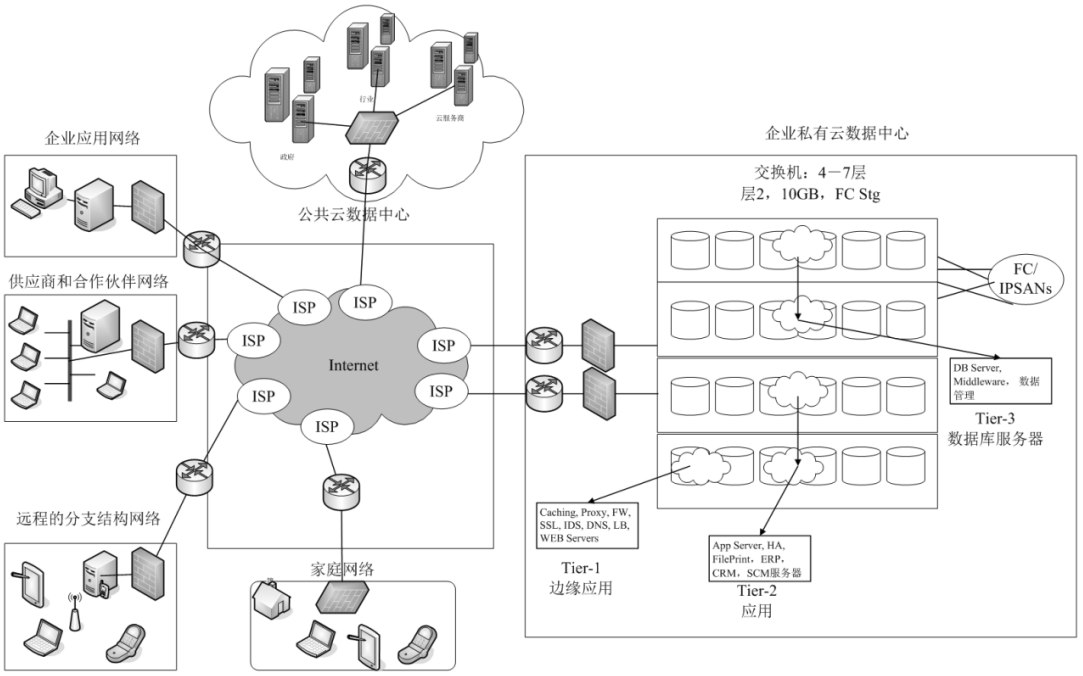
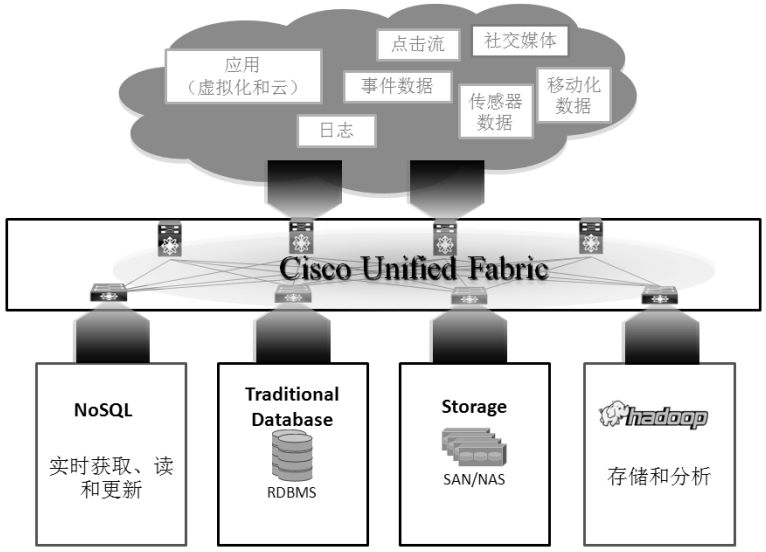


图 4-18　典型的云计算环境下的企业级网络架构

为了进一步满足大数据应用持续的要求，需要对现有企业网络架构进行升级，思科公司提出的统一的以太网结构（Unified Ethernet Fabrics，UEF）或正在兴起的软件定义网络（SDN）是解决这个问题的技术趋势。

1. 统一的以太网结构

统一的以太网架构（UEF）正快速发展，它很适合云计算和大数据的需求。UEF 是一个更扁平和集中的网络，它是架构在各种网络设备上的一个虚拟化网络平台，如图 4-19 所示。UEF 的特点如下。



资料来源：CISCO公司

图 4-19　思科公司统一的以太网架构

- 集中的网络架构。减少了网络设备的复杂性，以及与多个 Fabrics、分开的网络适配器和布线相联系的大量成本花费。
- 网络扁平化。网络架构的扁平化设计则最大化地提升了网络效率、减少了拥塞，并通过产生用于负载均衡和冗余的第二层网络路径，解决了扫描树的限制。
- 虚拟化。UEF 通过虚拟底盘的体系架构，统一了多个交换机的访问，逻辑上这些设备被当作一个设备来管理。这就产生了虚拟交换机的资源池，免除了手动配置的必要。这个设计提供了任何设备延迟的可预测的大数据集群服务器之间的流量带宽。
- 多个路由路径选择。通过利用通过网络的多个路径并连续决定最有效的路由，UEF 能实现全链接的利用。
- 可靠性。UEF 带来了分布式网络，对失效更有弹性和容错能力。

2. 软件定义网络（SDN）

与 UEF 的理念类似，但软件定义网络（SDN）更着眼于作为下一代企业网络的架构，

具有更好的开放性，并且得到更多业界的支持。

软件定义网络（SDN）能够将网络控制从物理基础设施中解耦出来，通过软件和虚拟化在更加全局的角度对网络设备进行控制，如图 4-20 所示。不同的网络设备通过开放的接口来进行整合，如 OpenFlow，一个可扩展的、可能是开源的网络操作系统架构在 OpenFlow 交换机上，通过很好定义的 API 实现网络操作系统对应用的支撑。SDN 和 OpenFlow 标准被认为是网络领域中的重要发展趋势，它们已经成为了谷歌、Facebook 和雅虎等云服务提供商和大型网络公司简化或自动化网络配置的一种主流趋势。用户不再需要手动操作网络中的任何交换机或路由器，即可快速添加和配置更多的网络功能。

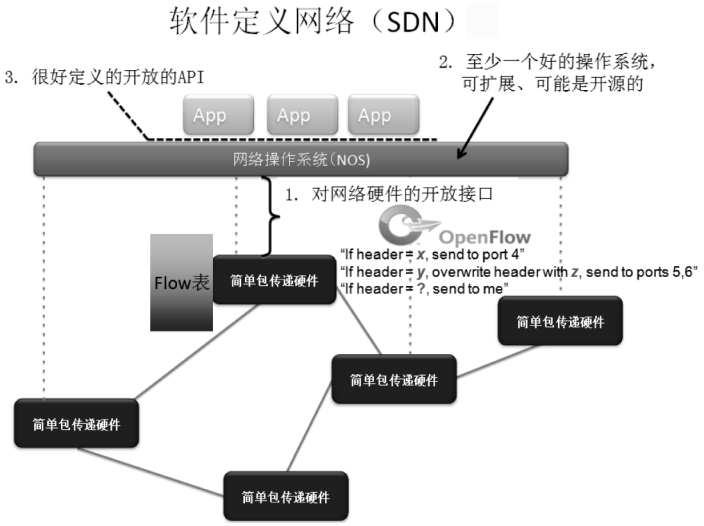


图 4-20　软件定义网络体系架构

网络架构从原理上将分为两个平面（Plane），数据面和控制面，如图 4-21 所示，其中，数据平面处理和交付网络上传输的数据包，控制平面则监控路由器等状态，所以关注点并不相同。数据平面与控制平面的比较如表 4-3 所示。

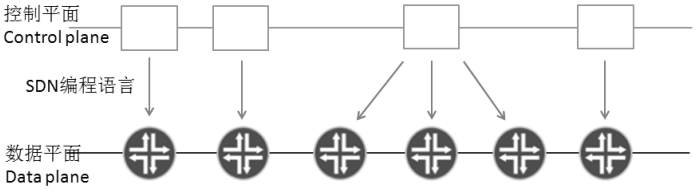


图 4-21　数据平面和控制平面

表 4-3　数据平面和控制平面的比较

	数 据 平 面	控 制 平 面
关　注	基于路由器和端点的状态	决定数据包怎样向前传递，以及传输到哪里
参　数	IP、TCP、Ethernet 等	路由、流量工程和防火墙状态……
目　标	缩短每个包的时间跨度	放慢每个控制事件的时间尺度

Open Networking Foundation（ONF）是一个标准组织[5]，主流网络设备公司都参与其中。OpenFlow 是其提出的。OpenFlow[6]类似于网络上的 X86 指令集，对像“黑盒”一样的网络节点提供开放的接口。图 4-22 是 OpenFlow 体系架构，它的特点如下。

- 分离控制平面和数据平面。
- 一个 OpenFlow 交换机的数据路径：包含一个流表（Flow Table），以及与每个流表项（Flow Entry）相关联的一个动作。
- 一个控制路径：包含一个控制器，它给流表中的流表项编程序。
- OpenFlow 基于一个以太网交换机，具有内部的流表，以及一个标准化的接口来增加和删除流表项。

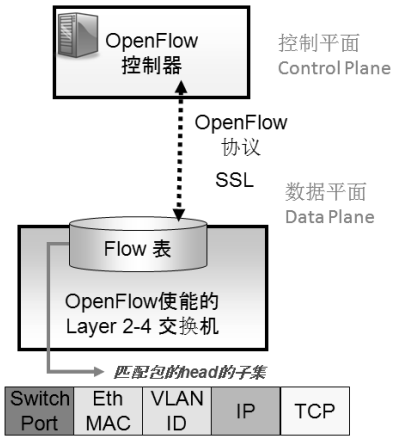


图 4-22　OpenFlow 体系结构

[5] www.OpenNetworking.org。

[6] http://OpenFlowSwitch.org。

斯坦福大学对 OpenFlow 有很大贡献，OpenFlow 的构架包括 5 层，底层是 OpenFlow 交换机、上一层是分片软件，其上是控制器，再往上是 OpenFlow 应用，最上面是监控和调试工具，如图 4-23 所示。

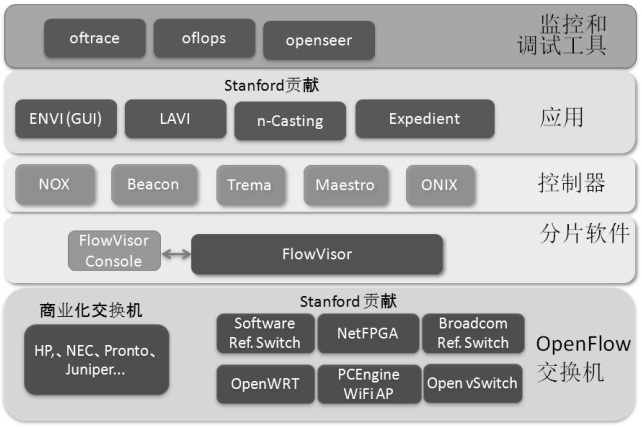


图 4-23　OpenFlow 模块

在企业级应用中，HP 等公司提出的基于 OpenFlow 的 SDN 架构可以分为 3 层，分别是：基础设施层，在这一层，网络设备通过 OpenFlow 的可编程接口实现连接；控制层，通过 SDN 的控制器来实现控制平面的功能；应用层，通过开放的应用 API 来实现 SDN 对应用的网络支撑，如图 4-24 所示。

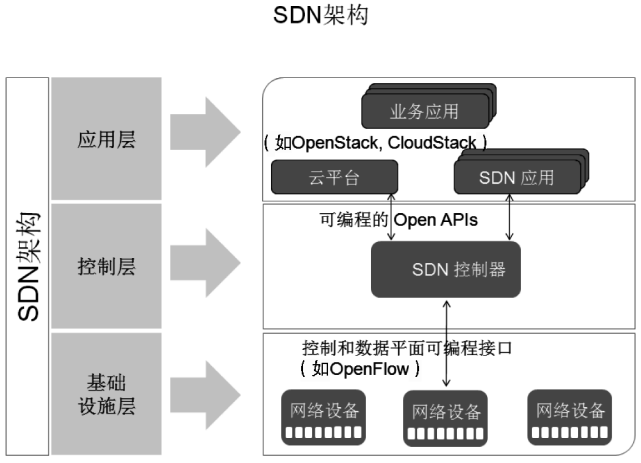


图 4-24　企业级 SDN 架构

OpenFlow 和 SDN 的真正价值源自其能力。

- 控制平面与数据平面分离，使得分布式系统得到集中控制。
- 开放的数据平面控制协议，使得交换设备商品化，避免厂商锁定，使得企业通过 HP、Big Switch、NEC 和 NTT 等厂商的控制器，以及 NoX、Beacon、Floodlight 和 Trema 等厂商的开源控制器实现互操作性。
- 网络设备虚拟化允许企业与多个机构按照不同属性来相互共享通用的物理基础设施，并实现路由重定向、负载均衡、提供峰值需求的带宽。
- 开放的控制平面管理接口，网络管理员能够使用网络 API 来编写软件。
- 可编程性使得其适用于不同使用场景的网络虚拟化应用和相应的应用粒度。例如对企业中所涌现的自带设备（BYOD）等现象进行管理等。
- 扩展和隔离不同数据中心或云租户网络的策略，将付费用户与内容、服务相连接等。

SDN 首先提供了这些能力，然后是拓展在其上运行的各种新的网络应用：如网络虚拟化、自带设备（BYOD）应用策略等。SDN 在企业内部的应用刚刚开始，它将是未来支撑大数据应用的基础网络架构。

4.4　大数据与虚拟化技术的整合

虚拟化是一个广义的术语，是指计算元件在虚拟的基础上而不是真实的基础上运行，是一个为了简化管理和优化资源的解决方案，如图 4-25 所示是传统技术架构与虚拟化技术架构的比较。虚拟化技术提供给应用系统一个集中的、共享的资源池，并且屏蔽了计算资源的异构，对用户来说，在计算池中的所有计算资源都是标准化的桌面、存储、服务器和网络虚拟设备，计算能力能够被“服务化”。这使得计算资源的可用性大大增强，计算资源能被统一部署、管理、动态调配和弹性收缩，避免计算资源的搁置和浪费。因为采用标准化的计算能力，可以实现管理自动化和按需的自服务，并降低计算设备的投入成本。因此，近年来虚拟化技术已经成为支持云计算和大数据应用的主流技术。

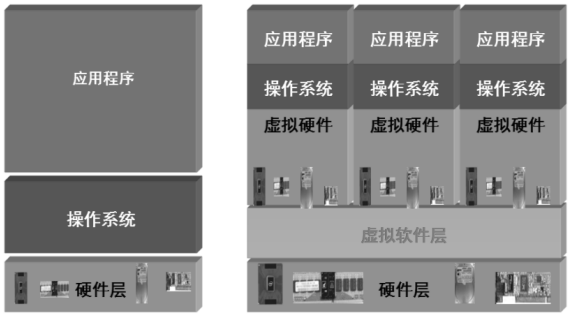


图 4-25　传统技术架构与虚拟化技术架构的比较

虚拟化技术能构建支持异构存储、异构网络、异构主机及各种不同版本应用的统一的计算资源池的架构，实现存储虚拟化、网络虚拟化、主机虚拟化等。如图 4-26 所示为典型的虚拟化架构。

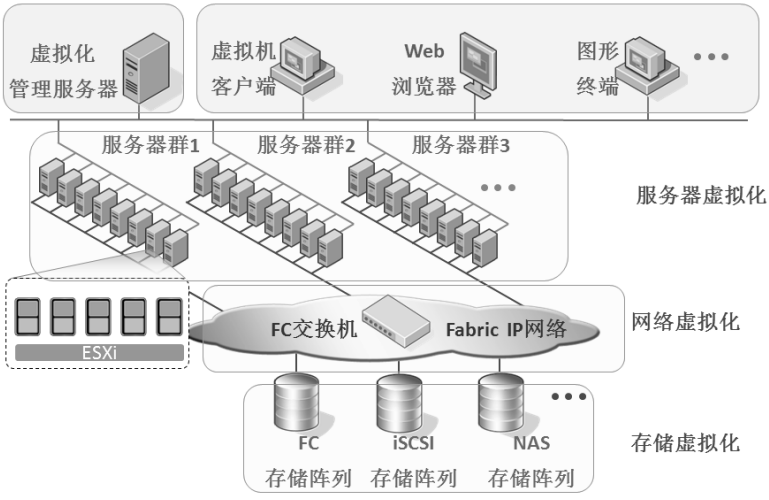


图 4-26　典型的虚拟化架构

存储虚拟化，可以形成统一的存储池，屏蔽各个存储设备的异构，实现阵列高可用，以及在线数据迁移等。对于大量的非结构化的数据的存储，通过存储的虚拟化网关，用户不再关心文件存储的路径，通过单一位置提供的文件名就可以访问。存储虚拟化是构建集群存储的基础，能够支持实现海量数据的动态分级存储。

网络虚拟化，可以将两台或多台设备虚拟为一台设备，实现统一转发、统一管理，并实现跨设备的链路捆绑，简化网络协议的部署，大大缩短设备和链路收敛时间（毫秒级），以链路负载分担方式工作，利用率大大提升。网络虚拟化是实现 SDN 的基础技术。

主机的虚拟化技术，可以实现“一分多”，即将一台服务器虚拟成多台虚拟机，在进行 Hadoop 平台的安装和实验阶段，可以采取这样的方法来进行。主机虚拟化技术也可以实现“多合一”，将多台服务器虚拟成一个虚拟服务器，实现不同大数据计算集群的统一关联和资源共享。如图 4-27 所示为用虚拟化技术统一大数据平台，可以使得计算资源池能够按需求更快、更容易地提供新的数据集群，允许工作负载的混合，利用虚拟机来提供隔离，基于虚拟的拓扑来优化数据性能，基于虚拟拓扑使得系统更可靠。

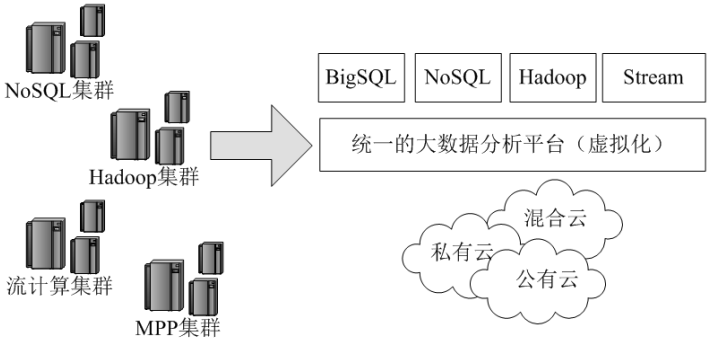


图 4-27　大数据分析平台（虚拟化）

桌面虚拟化可以通过瘦客户端或者其他任何与网络相连的设备来访问跨平台的应用程序，以及整个客户桌面。这是实现云计算、移动计算和带着设备工作（BYOD）的基础。

4.5　在云计算平台上的大数据云

云计算是基于网络的 IT 服务交付模式，它借助分布式计算、并行计算、虚拟化、负载均衡和统一管理等技术手段，将存储、网络、主机、应用等计算能力融合为无形的 IT 资源和 IT 服务能力，并实现 IT 服务的按需交付。Hadoop 分布式计算平台是在大数据处理和分析方面云服务的重要支撑技术。

作为计算模式创新，云计算使“一切都可成为服务”，这种服务可在任何的云终端获取，称为云服务。常见的云服务有基础设施作为服务（IaaS）、平台作为服务（PaaS）、软

件作为服务（SaaS）、业务流程作为服务（BPaaS）、数据中心作为服务（DaaS）等，如图 4-28 所示。云计算包括公有云和私有云，对一些大型企业来说，目前核心应用采取的是私有云。更多的中小企业则可以利用云服务商提供的公有云服务。

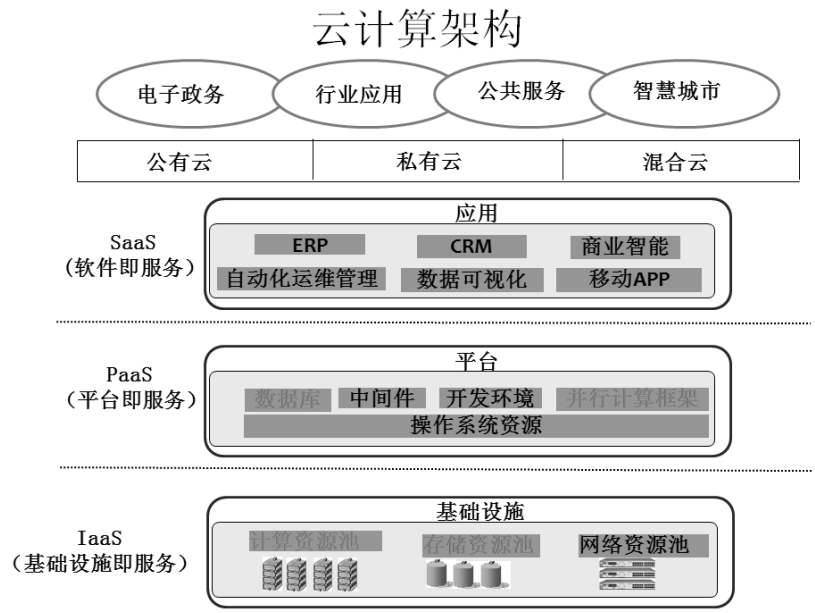


图 4-28　云计算架构

基于云计算的基础架构，可以构建大数据云，大数据云服务包括基础设施级、数据基础设施级、数据平台级、开发级和分析级的云服务，如图 4-29 所示。Google 正在建设在云中分析大数据的服务，Google 的这项服务被称为 BigQuery[7]。该服务让开发者可以使用 Google 的架构来运行 SQL 语句对超级大的数据库进行操作。BigQuery 允许用户上传他们的超大量数据并通过其直接进行交互式分析，从而不必投资建立自己的数据中心。Google 曾表示 BigQuery 引擎可以快速扫描高达 70TB 未经压缩处理的数据，并且可马上得到分析结果。亚马逊的弹性 MapReduce 服务也是基于云的 Hadoop 服务。

[7] https://developers.google.com/bigquery/what-is-bigquery。

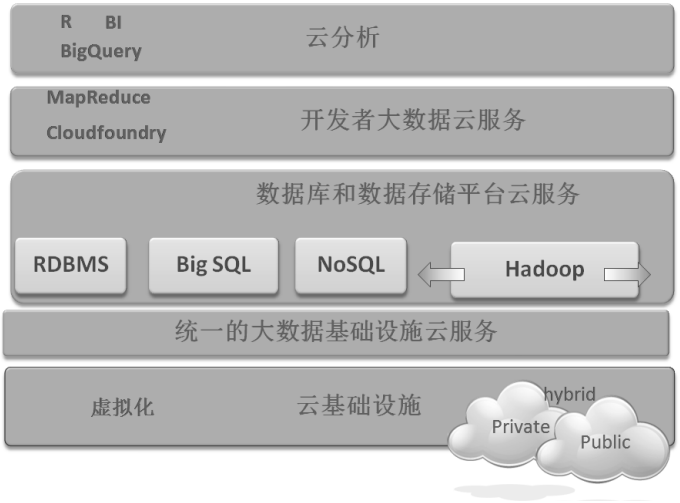


图 4-29　大数据云服务

4.6　大数据与信息安全

随着数据数量和种类的增多，企业面临的风险更加复杂，传统的数据保护方法常常无法满足。大数据意味着数据及其处理平台的分布式和健壮性，由于平台可靠性和容错性增强，单点被攻击破坏对整体平台的影响较小。但另外一个方面，在海量数据中这种单点的攻击容易被忽略，导致寻找攻击点的难度加大。而且，大数据牵扯面广，系统遭受的攻击一旦蔓延开来，传播速度非常快。大数据对信息安全提出了很多新的课题[8]。

1. 分布式编程框架中的安全计算。
2. 非关系型数据存储的安全最佳实践。
3. 安全的数据存储和事务日志。
4. 终端输入的确认/过滤。
5. 实时安全/合规监控。

[8] https://cloudsecurityalliance.org/research/big-data/。

6. 可扩展的、可组合的隐私保护分析。

7. 强制加密的访问控制和安全通信。

8. 粒度访问控制。

9. 粒度审计。

10. 数据保护与容灾。

这 10 个方面的大数据安全面临的核心问题、威胁与挑战、当前措施如表 4-4 所示。

表 4-4 大数据安全

	核心问题	威胁和挑战	当前措施
1. 分布式编程框架中的安全计算	怎么保护分布式并行计算框架的安全	• 计算工作节点的故障 • 敏感数据的访问 • 输出信息的隐私权	• 信任建立：初始化、周期性信任更新 • 强制的访问控制 • 隐私保护的变化
2. 非关系型数据存储的安全最佳实践	怎么保护过去没有建立安全意识的非结构化数据的安全	• 缺少严格的认证和授权机制 • 缺少计算节点间的安全通信	• 通过中间件层的加强 • 密码必须用暗码 • 休眠数据的加密 • 用 SSL/TLS 保护通信
3. 安全的数据存储和事务日志	怎样保证大数据存储管理的安全性	• 数据的保密性和完整性 • 可用性 • 一致性 • 合谋（Collusion）	• 加密和签名 • 数据所有权的证明 • 周期性的审计和哈希链 • 基于加密的策略
4. 终端输入的确认/过滤	怎样信任来自不同终端，如传感器、设备和应用的数据	• 对手可能损害设备和软件 • 对手可能克隆伪造设备 • 对手可能直接控制数据源 • 对手可能在传输过程中损害数据	• 损害验证软件 • 信任证书和被信任的设备 • 对外来者侦查的分析 • 密码协议
5. 实时安全/合规性的监控	怎么利用大数据的分析来帮助改进系统的安全性	• 基础设施的安全 • 监控代码自身的安全 • 输入源的安全 • 竞争对手造成的数据毒害	• 安全基础设施结构措施 • 安全代码实践 • 输入源安全措施对外来者侦查的分析
6. 可扩展的、可组合的隐私保护分析	怎么利用大数据的分析来帮助改进系统的安全性	• 发现主机的脆弱性 • 内部攻击 • 对不信任的合作伙伴的外包分析 • 通过数据共享的无意的泄露	• 休眠数据的加密、访问控制和认证机制 • 清晰的责任界定原则和数据集访问的日志制度 • 再签定和隐私多重性的意识

续表

	核 心 问 题	威胁和挑战	当 前 措 施
7. 强制加密的访问控制和安全通信	怎么加强端对端的数据保护	• 强制的访问控制 • 搜索和过滤 • 计算的分包 • 数据的完整性和匿名的保护	• 基于身份的加密和基于属性的加密 • 支持搜索和过滤的加密技术 • 完全的同态加密 • 用可信赖的第三方的组签名
8. 粒度访问控制	怎么控制对广泛的数据集的访问	• 持续跟踪独立的数据元素的保密性 • 在分析的变化过程中维护访问标签 • 持续跟综用户的角色和授权	• 采取合适的类度级别，如行的级别、列的级别、单元的级别 • 最少要遵守访问限制的栅格，正在研究更复杂的数据变化 • 认证、授权和强制的访问控制
9. 粒度审计	怎么审计种类繁多、分布式的系统	• 审计信息的完备性 • 适时访问审计信息 • 审计信息的完整性 • 审计信息的授权访问	• 基础设施解决方案 • 扩展安全信息和事件管理（SIEM）工具
10. 数据保护与容灾	怎样跟踪复杂的元数据	• 安全的数据集合 • 数据和元数据的一致性 • 内部威胁	• 认证技术 • 消息摘要 • 系统级的访问控制和密码体系

在大数据应用体系架构中，数据采集（日志和物联网数据采集）、数据处理、数据存储和数据分析利用的不同阶段，信息安全的关注点不同。在数据采集阶段，关注终端输入的确认和过滤、数据保护，物联网上的机器数据采集关注粒度的访问和控制、粒度审计等。在数据处理阶段，则关注分布式计算框架的安全性、安全事务、实时安全等。在数据存储阶段，关注实时安全、安全通信、粒度访问等。在数据分析阶段，数据保护和容灾更为重要。如图4-30所示为大数据安全的关注点。

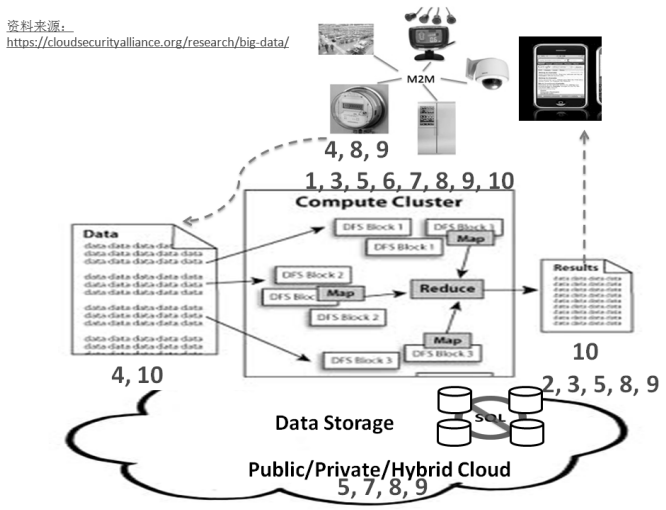


图 4-30　大数据安全的关注点

4.7 以银行客户分析为例，分析一个大数据的平台整合

如图 4-31 所示为某银行的大数据应用架构，是 A 银行“基于大数据的新一代银行客户分析系统”的整体技术应用和整合架构。

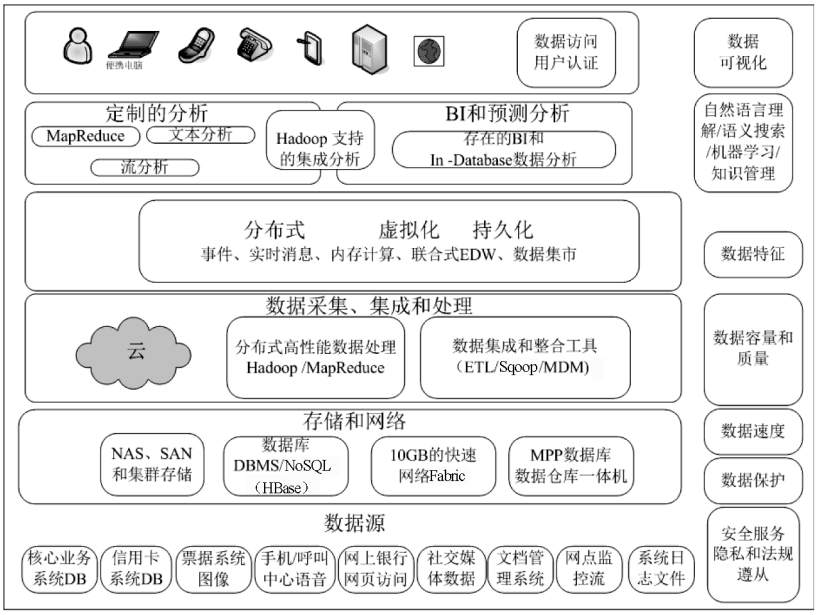


图 4-31 某银行的大数据应用架构

数据采集和数据整合是很重要的部分，数据从各个已有业务系统采集，有些系统本来就记录和存储了大量数据，如电话语音、系统日志等，另外一些系统需要增加数据采集的支持。Hadoop 平台和传统数据库系统之间的数据采集通过 Sqoop、ETL 等技术来实现。

A 银行原有存储系统和数据仓库一体机（MPP）等存储和分析的硬件资源十分充足，能够通过虚拟化技术提供存储给大数据处理和分析。更低成本的 PC 服务器集群和刀片机是 Hadoop 平台的硬件基础，能够很大程度实现成本节约，可利用的资源也很丰富。商业化的大数据机做了很多优化，需要对成本和性能进行比较后，在今后应用中做出选择。网络架构满足当前大应用需求，并没有做调整。目前，A 银行仍然以私有云应用为主，基于大数据的新一代银行客户分析系统也部署在总行的私有云架构中。

随着大数据应用的不断扩展，特别是非结构化数据的分析，存储和网络升级的方向是集群存储和 SDN。

第 5 章

大数据应用的实践方法与案例

本章阐述企业如何启动大数据应用的项目，并介绍企业级应用的典型案例。

5.1 实践方法论

大数据应用的实践流程是从识别业务要求和评估数据分析能力开始的，在业务战略指引下，组织从现有的和新的数据来源中获取新的洞察力，制定大数据应用战略，并规划大数据架构、关键技术、数据源和分析方法，以支持业务战略落实。大数据应用的实施中采取先试点后推广的策略，随着时间的推移逐步地实施和升级相应的大数据架构，最终落实大数据应用，支撑企业的战略决策。大数据应用实践的流程和实施方法如图 5-1 所示。

5.1.1 业务需求定义

成功的大数据项目通常是以使命和目标开始的，这是与业务战略目标一致的、具体的、很好被定义的大数据应用的使命和目标。为了明晰目标，需要有一个大数据的业务需求定

义，解决“为什么要做”、“做什么”和“做成什么样”的问题。毕竟，大数据平台是一个新的、还未曾普遍使用的技术平台，需要通过业务需求定义做出系统性的分析，以发现将来可能提出的更多业务需求。

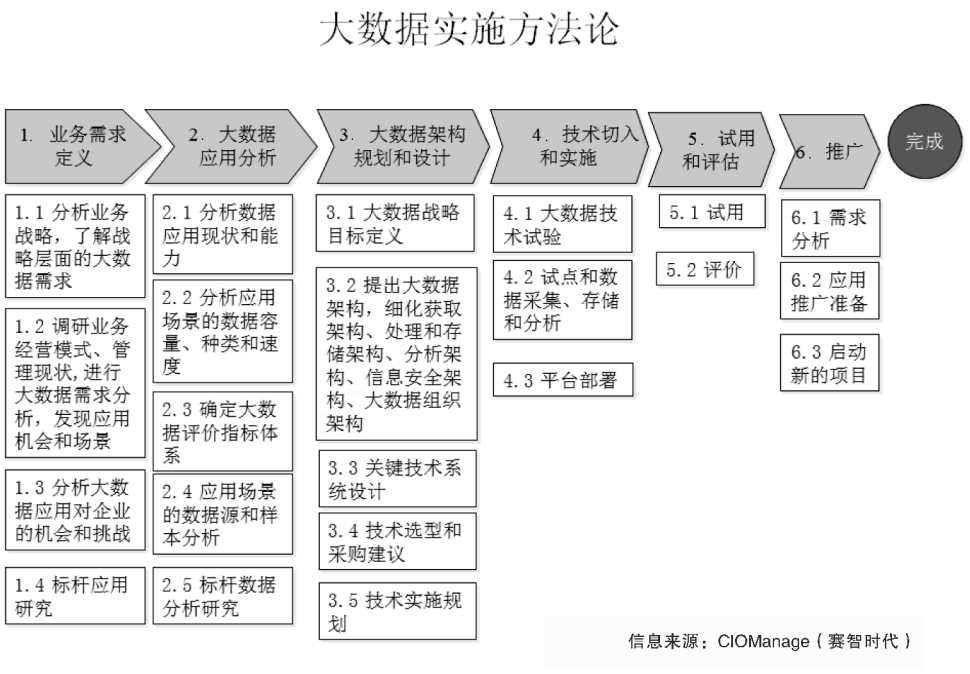


图 5-1　大数据应用实践的流程和实施方法

业务需求定义阶段要分析组织的业务战略，总结对战略决策具有关键性意义的分析数据需求，通过业务部门、客户和合作伙伴的调研和分析，根据业务优先级确定分析能力的优先级，确定逐步建立分析能力的策略。要进行大数据的 SWOT（优势、劣势、机会和挑战）分析，分析定义大数据的机会、关键业务和使命的挑战。要调研初始的用例或用例集，业务需求满足程度以及大数据能够交付的价值。

主要关注点包括如下内容。

- 找到各种应用机会，这些领域通过大数据能够更好地服务客户，完成业务使命和减少成本。
- 找到关键的业务挑战，解决这些业务挑战的大数据应用的潜在应用场景。

- 评价大数据是不是能够做出一个独一无二地满足应用场景需求的承诺。
- 找到大数据投资与传统分析投资，或者干脆什么都不做进行比较的价值。
- 创立一个战略远景，并分解这个远景为几个战术的阶段目标。
- 不要试图在初始项目中用大数据解决所有问题，要着眼于实际的行动，但对关键业务需求要用战略的眼光。

5.1.2　数据应用现状分析与标杆比较

从现有数据开始做分析，并与业界标杆进行比较，确定出差距，着眼企业数据利用的基础来确定近期的目标。从内部着眼，允许企业利用现有数据、软件和技能，提供近期业务价值，并且在考虑扩展现有能力而处理更复杂的数据来源和类型之前积累重要的经验。大多数企业希望通过这样做而充分利用现有数据库中的信息，同时扩展其数据仓库，以处理更大数量和更多类型的数据。主要关注点包括如下内容。

- 评估组织当前的数据和技术能力，评价当前的数据和技术能力是不是能够满足定义的业务需求和应用场景。
- 评估应用场景在数据速度、种类和规模方面的需求，确定是否需要升级到大数据的项目，还是传统方法就可以满足需求。
- 评估需要满足定义的应用场景的数据和数据源，评价数据当前的可用性情况。
- 评估支持访问、治理、管理和分析数据的技术需求，以及与当前能力的差距。
- 评估组织的分析能力。
- 分析最佳实践和参考架构。
- 利用参考架构、业界最佳实践标杆来找到差距。
- 开发一个投资回报率（ROI）评估指标来评价当前的部署阶段是不是具有价值（在政府部门，ROI 只是一个逻辑概念，ROI 可能是对公众的更好的服务，而不必是一个财务的 ROI）。

表 5-1 列出了评估分析能力的各项内容，表 5-2 列出了最佳实践分析的视角。

表 5-1　评估分析能力

能力 （自高而低）	信 息 管 理	分析技术和工具	面向数据的文化
刚有热情	怎样管理特殊数据	使用电子表格和标准化的报告	分析是不是战略的组成部分
专业化	业务线有集中的数据管理	使用预测分析的工具和技术	被业务线的业务绩效评价所驱动
协作化	企业应用集成在进行	使用计分卡 使用仪表盘 实施了数据仓库和 BI	被企业范围的关注所驱动
重大变革	集成化的、简化的信息平台	强大的分析工具和能力的集合	分析驱动的组织

表 5-2　最佳实践分析的视角

内　　容	细　　节
主数据的单一视图	标杆企业在生成企业级的主数据模型方面有哪些好的做法？如何建立统一的客户、产品、资产、资金、组织、人员、渠道等主数据的视图？如何支撑大数据分析
业务流程管理	标杆企业如何通过业务流水线和业务流程管理来实现业务数据的采集、分析和改进
信息管理	标杆企业如何管理企业在各个环节产生的关键信息？如何利用历史系统中的数据？如何获取与客户沟通中有价值的信息
整合和简化信息集成和数据分析	标杆企业如何整合和简化集成和分析的环境？如何整合新的数据分析的技术
整合和优化基础设施平台	标杆企业如何整合和优化信息化基础设施？如何实现不同平台之间的信息共享？如何保护历史基础设施投资

5.1.3　大数据应用架构规划和设计

大数据应用架构规划和设计阶段，要以业务价值为基础、以分析战略为驱动、以灵活性和扩展性为原则，设计一个将来的大数据技术应用架构。主要关注点如下。

- 规划整个企业的大数据战略，战略规划包含企业内的大数据愿景、战略目标和要求，它对于在业务用户的需求与 IT 实施路线图之间做到协调非常关键，它实现了关于企业如何利用数据和分析驱动的战略来改进业务目标的一致理解。
- 规划大数据平台架构，包括采集、处理、存储和分析的模块规划，以及实现该架

构所需数据、工具和硬件，从而定义了企业内大数据的建设和实施范围。

- 设定平台切入点，找到成功输出的基本原则。
- 在选择的应用场景或者多个应用场景的支持下，开发体系架构的路线图。
- 规划开发、测试和部署的模式。
- 规划项目管理、设备选型和采购，考虑必需的技术、资源和团队成员。
- 规划有效的大数据治理的结构、工具和流程。
- 设计大数据管理政策、隐私保护和安全的措施。
- 制订试用计划，以降低业务和技术风险。
- 制订投资计划。

5.1.4　大数据技术切入与实施

大数据应用的技术实施阶段将按照设计方案进行系统采购、数据准备、系统集成、安装调试、技术开发、模型开发、系统整合、系统联调等工作。主要关注点包括如下内容。

- 实施步骤和方法工具。
- 技术切入点。
- 试验数据准备。
- 实施环境和配置管理。
- 系统验收指标。
- 被实施系统的灵活性和可扩展性，以扩展到支持随后的应用场景和相应的数据规模、种类和速度。

通常根据大数据的应用场景，选择不同的技术切入模式，如表 5-3 所示。

表 5-3　大数据实施技术切入点的选择

应 用 类 型	侧　重　点	切　入　点
Variety 优先型	此类大数据应用主要处理和分析非结构化数据	需要探索、理解和分析各种数据源，跨越结构化、非结构化、半结构化的数据，要求横向扩展，并保证高性能低成本，应以 Hadoop 或 Hadoop 类的技术作为切入点
Volume 优先型	此类大数据应用主要用于大容量数据的访问和分析	经常把数据库或者数据仓库体系结构作为切入点启动实施，在传统数据处理的基础上逐步扩展到更大规模分布式计算系统
Velocity 优先型	此类大数据应用主要用于数据流的快速分析和处理	倾向于把流计算作为一个切入点。

大数据实施环境的部署要点如下。

- 企业大数据技术平台的应用领域，例如客户分析、风险分析，还有产品创新。
- 目标应用场景的准备，例如针对客户分析的目标客户分析、客户流失分析等。
- 大数据应用部署规模，关键指标是平均集群规模、I/O 利用率等，如某银行平均集群规模为 50 个节点，而某大型图书馆的集群规模为 400 个节点。
- 硬件部署（服务器/核、存储、网络和管理工具），服务器集群还是大数据一体机，集群服务器的 CPU、内存、存储和配置，存储网络的配置，网络的配置等。
- 软件套件，商业化套件还是开源套件？哪些是套装软件？哪些是单独安装的软件？
- 性能瓶颈，如集群的节点间的延时、存储规模、SSD、网络 I/O 等。
- 高可用性，如何规避 HDFS 的 SPOF，如何做 NameNode 的高可用性（HA）。
- 对快照、镜像、完全随机读写文件系统的需求。

5.1.5　大数据试用和评估

组织开始利用现有的和新的数据源进行大数据技术系统的应用试点。一个典型的业务需求被用于试点，相应的真实数据被用于采集和分析，并设法做出第一次评估，评价大数据应用是否满足了设定的业务目标和技术性能指标。找出当前的技术实施与大数据应用参考体系架构仍然存在的差距，为项目的改进和持续实施提供依据。在试用过程中，还要第一次评估业务流程、政策、数据治理、隐私保护和安全等方面的满足程度。

评估一个持续的过程，需要不断审核部署的体系架构和技术是不是能满足组织的更广

泛的业务需求；需要评价大数据投资回报率如何；需要连续审核计划执行是否与数据治理、隐私保护和安全政策一致，审核大数据目标是不是与当前的政府和法律法规相背；通过持续不断地评估和反馈，持续地改进和优化大数据应用。

5.1.6　大数据应用推广

通过上述阶段的成功实践，组织会把在一个领域成功的经验推广到其他业务领域，建立起一个更强大、一致和企业级的大数据平台。例如，在客户分析领域将大数据的成功应用推广到风险分析和绩效分析领域。至此，企业级的大数据分析平台将开始提供跨多个业务部门和领域的分析能力，并能提供一组企业范围的公共服务来支持若干个即将启动的大数据分析项目。这些服务是建立在大数据核心技术架构基础之上的，主要包括如下内容。

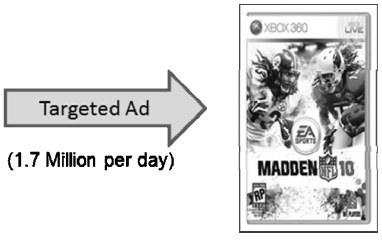
- 数据集成和数据治理。
- 隐私保护和数据安全。
- 公共元数据管理。
- 可视化和高级分析。

5.2　应用案例

5.2.1　亚马逊

亚马逊的一个案例如表 5-4 所示。

表 5-4　亚马逊的一个案例

应用场景	**目标广告和点击流分析：**通过客户行为分析来改进广告花费的回报。例如，用户最近购买了一个运动方面的电影，并正在搜索视频游戏。目标广告能主动给用户推荐一款游戏 

续表

数据规模	35 亿个记录 7100 万不同的 cookies 170 万目标广告/天
技术架构	采用亚马逊提供的基础设施云服务，后台是 Hadoop 技术架构的 100 个节点的按需弹性 MapReduce 集群。弹性 MapReduce 是一项能够迅速扩展的 Web 服务，运行在亚马逊弹性计算云（Amazon EC2）和亚马逊简单存储服务（Amazon S3）上。 ——处理时间从 2 天降到 8 小时 ——广告投资回报率增加了 500%

5.2.2 雅虎

雅虎的一个案例如表 5-5 所示。

表 5-5 雅虎的一个案例

应用场景	**分类：**将相似的新闻分类在一起。 **学习：**给用户反馈正确的知识
数据规模	20 000 000+知识 2 000 000+查询/天
技术架构	Hadoop+Mahout ——在 Hadoop 上每小时计算所有的新闻 ——在 Hadoop 和 Mahout 上找出知识和查询之间的关系

5.2.3 淘宝网

淘宝网应用案例如表 5-6 所示。

表 5-6 淘宝网应用案例

应用场景	大规模小图片的存储和读写。 （对于大多数系统来说，大规模的小文件存储与读取是一个棘手的问题，因为磁头需要频繁地寻道和换道，因此在读取上容易带来较长的延时。在大量高并发访问量的情况下，延时极其严重。）

续表

数据规模	整个淘宝网流量中，图片的访问流量会占到 90%以上。 整体图片存储系统容量 1800TB（1.8PB），已经占用空间 990TB（约 1PB）。 保存的图片文件数量达到 286 亿多个，这些图片文件包括根据原图生成的缩略图（一个原图可能要生成 20 个不同尺寸的缩略图）。 平均图片大小是 17.45KB，8KB 以下图片占总量的 61%，占存储容量的 11%。 淘宝网的图片文件数量以每年 2 倍的速度增长
技术架构	TFS（淘宝文件系统，Taobao File System）2007 年正式上线运营。在生产环境中应用的集群规模达到了 200 台 PC Server（146G*6 SAS 15K Raid5），文件数量达到上亿级别，系统部署存储容量为 140 TB，实际使用存储容量为 50 TB，单台支持随机 IOPS 200+。到 2009 年 6 月，TFS 1.3 版本上线，集群规模大大扩展，部署到淘宝的图片生产系统上，整个系统已经从原有 200 台 PC 服务器扩增至 440 台 PC Server（300G*12 SAS 15K RPM）+ 30 台 PC Server（600G*12 SAS 15K RPM）。支持的文件数量也扩容至百亿级别，系统部署存储容量为 1800TB（1.8PB），当前实际存储容量为 995TB，单台 Data Server 支持随机 IOPS 900+

5.2.4　Facebook

Facebook 案例如表 5-7 所示。

表 5-7　Facebook 案例

应用场景	大数据分析
数据规模	系统的数据量是 15PB（压缩以后为 2.5PB） 每天增加的数据量是 60TB（压缩以后是 10TB）
技术架构	在 Facebook 数据分析系统中，关系型数据库系统处在系统的边缘（挂接在 Web Server Farm 上），负责进行 OLTP 类的事务处理。事务数据通过定时的装载，导入核心生产用的 Hive-Hadoop 集群系统（Hive-Hadoop cluster），重要的分析功能在 Hive 系统里面完成。经过分析和聚集的结果，可以重新注入关系型数据库系统（包括 Oracle RAC，federated MySQL 等），接受用户的查询。 为了减轻即席查询对核心 Hive 系统的压力，数据被复制到一个备份的 Hive-Hadoop 集群系统（ad hoc Hive-Hadoop cluster），进行用户即席查询的处理，隔离未经优化的查询有可能给核心 Hive 系统造成的性能冲击，保证核心数据分析系统的性能。真正的复杂

续表

	深度分析，依靠高度可扩展的 Hive-Hadoop 集群系统来完成。由于 MapReduce 技术所具有的良好的扩展性，可以实现大量历史数据的在线，对历史久远的数据也可以很快地进行分析，结合新数据和新算法，有利于新知识的发现 Facebook 的 Hive 应用架构如图 5-2 所示

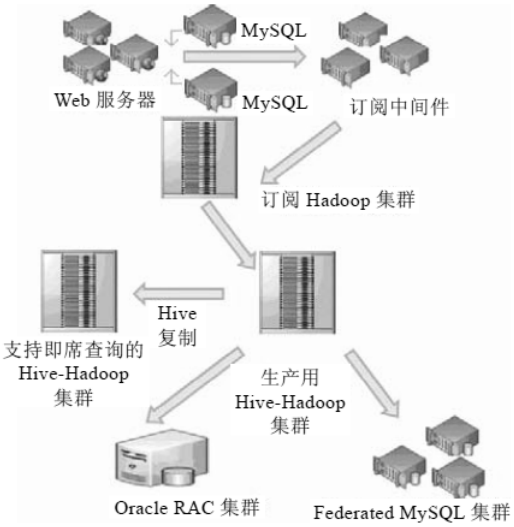


图 5-2　Facebook 的 Hive 应用架构

5.3　以银行客户分析为例的实施案例分析

5.3.1　银行基于大数据的客户分析的业务需求

A 银行“基于大数据的新一代银行客户分析系统项目”，以业务需求分析起步，分析外部的经济环境、严格的监管要求、多样性的客户需求以及激烈的市场竞争中带给 A 银行的业务挑战，如个人客户的流失、大企业客户的破产等。

分析 A 银行业务发展战略，调研以客户为中心、高绩效、全风险管理、流程银行、特色服务等业务实践，以及数据大集中、电子银行、管理信息化等信息科技实践，发现 A 银行的数据分析的各种应用机会，例如客户个性化特色服务、客户流失预测、客户价值分析等。

选择关键业务应用场景，确定 A 银行客户大数据分析项目的业务战略目标，既有长远的目标，如“高洞察力的智慧银行”，也有具体的近期目标，如根据客户喜好提供特色服务等。

业务需求分析阶段的一项工作是做数据分析模型的设计和算法的选择，这是很重要的一项工作，也是后续大数据分析的基础。如图 5-3 所示为银行基于大数据的客户分析中的业务需求分析。

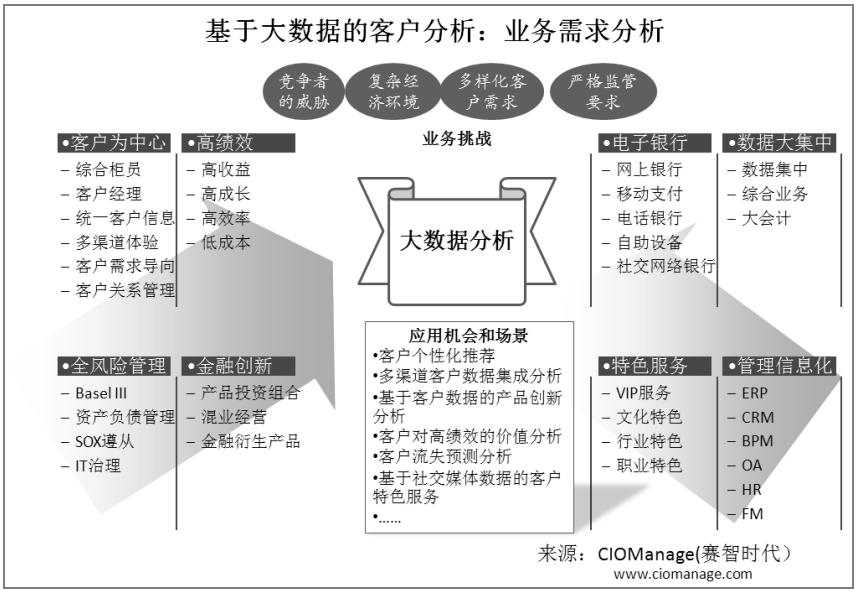


图 5-3　银行基于大数据的客户分析——业务需求分析

5.3.2　银行基于大数据的客户分析的现状与标杆比较

第 2 阶段，要评价 A 银行的数据分析需求是不是能够通过现有的数据分析能力来满足。调研 A 银行在综合业务系统中的客户分析功能和应用现状，在网上银行、信用卡等系统中的客户分析功能和应用现状，客户关系管理（CRM）系统的客户分析和报表，数据仓库（DW）和商业智能（BI）应用情况，评价这些系统是不是能够满足大数据关键应用机会和场景？存在哪些问题和不足？判断大数据应用是不是能够解决这些问题？

这个阶段的对标比较研究是要分析国内外银行在大数据领域的最佳实践，A 银行会对

标花旗银行、美国银行、汇丰银行、中国工商银行等行业标杆，分析和学习它们在大数据领域的应用实践，也可以参考行业研究报告[1、2]。例如，在美国银行，大数据就被用于客户行为数据跟踪和分析，主要包括网站点击、交易记录、银行家的笔记、电话银行的录音，根据这些数据美国银行来为客户提供特色服务，如预先性的、有竞争的信用额度，以及有选择性的 BankAmeriDeals 现金返回服务。如图 5-4 所示为银行基于大数据的客户分析中的数据分析的现状。

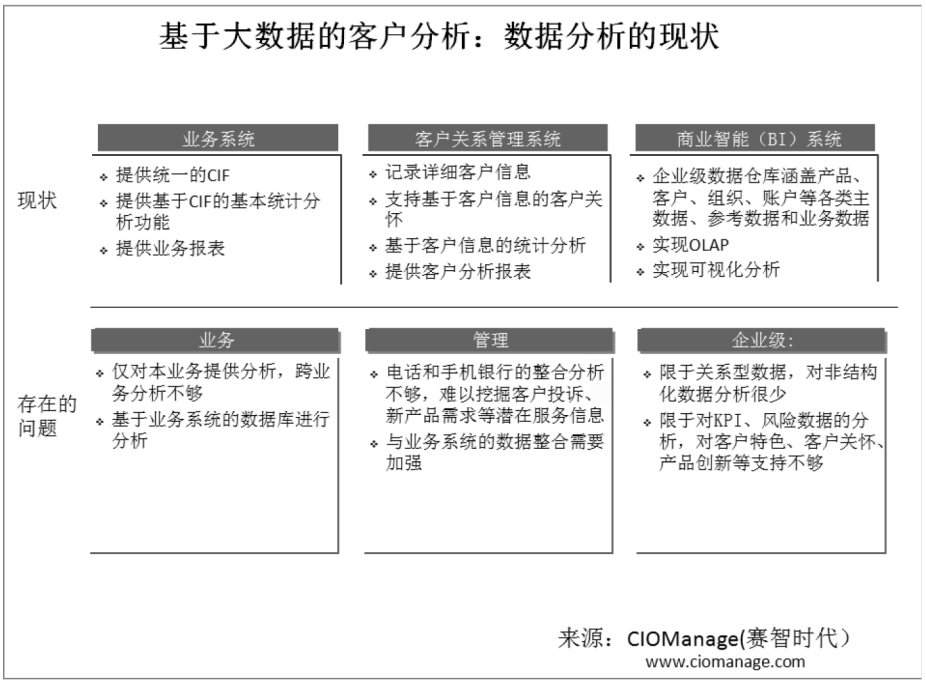


图 5-4　银行基于大数据的客户分析——数据分析的现状

5.3.3　银行基于大数据的客户分析的应用架构规划与设计

根据 A 银行现有的数据分析架构和整个企业的 IT 架构，设计和规划 A 银行基于大数据的客户分析系统架构。当然，大数据不仅仅应用于客户行为分析，在风险管理、企业绩

[1] http://www-935.ibm.com/services/multimedia/Analytics_The_real_world_use_of_big_data_in_Financial_services_Mai_2013.pdf。

[2] 《2013—2014 年中国银行业大数据应用研究报告》，CIOManage（赛智时代），2013。

效等方面也有广泛应用。因此，银行大数据平台架构应支持不同类型应用的扩展。本书图 4-31 的某银行的大数据应用架构中即是 A 银行的大数据应用架构。基于 Hadoop 平台对非结构化数据进行客户分析正是 A 银行在大数据应用领域的切入点，典型应用是客户个人化特色推荐和流失客户预测等。

5.3.4　银行基于大数据的数据分析的实施、试点和推广

A 银行大数据分析平台选择了成熟的 Hadoop 发行版，包含了主要的商业化的 Hadoop 堆栈。

A 银行大数据的试点数据直接从网上银行网站点击记录、业务系统交易记录、系统日志、电话银行的录音等进行装载，使用 Sqoop 等装载工具。

试点过程中，A 银行采取了 3 种方式进行数据查询分析。一是用原有关系型数据库，用 ETL 工具从文档库和 Hadoop 中抽取数据到 Oracle 数据中，然后在 Oracle 数据库中用 SQL 工具做查询和分析。二是用 MapReduce，用 ETL 工具把数据从文档库和 Oracle 数据库中抽取到 Hadoop，然后与 MapReduce 一起生成所需要的分析。三是用 Hive，用 ETL 工具把数据从文档库和 Oracle 数据库中抽取到 Hadoop，文档必须做平整和模型化，但是 HiveSQL 是有限的，查询花很长时间，且 BI 支持有限。A 银行正在探索试用 Impala 来进行数据查询分析。

在数据高级分析和可视化方面，很多工作是基于已经部署的 SAS 系统的，A 银行也尝试基于 Mahout 来使用机器学习的算法，用于客户流失的预测。

在基于大数据的客户分析成功应用的基础上，大数据应用平台能进一步提供对风险分析、信用评级、绩效管理等领域的应用推广。

第 6 章

大数据应用的主流解决方案

本章阐述大数据产业链和大数据应用主流厂商的解决方案。

6.1 产业链

6.1.1 国际上的大数据生态环境

大数据的厂商生态图发布在投资人 Dave Feinleib 的博客[1]上，比较清晰地介绍了国际上主流的大数据研究、产品和服务厂商，本书引用了 Dave 的分析，如图 6-1 所示。当然，新的产品和厂商每天都在不断涌现，真正的检验还将来自市场。

大多数厂商的产品和技术都建立在 Apache 开源的 Hadoop 分布式计算和存储的基础支撑平台上，包括开源的 Hadoop/MapReduce、HBase、Mahout 和 Cassandra 等。

[1] http://blogs.forbes.com/davefeinleib/。

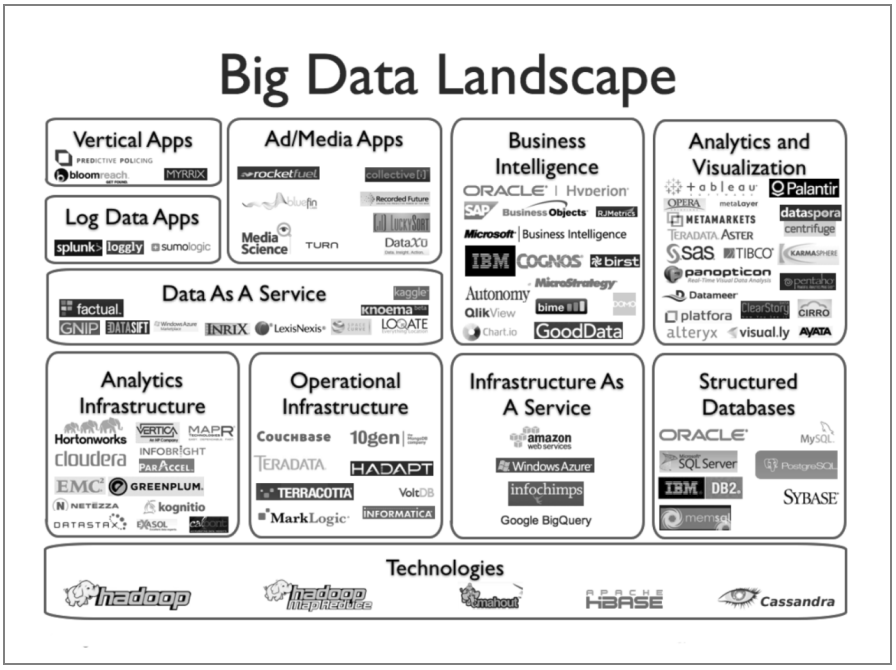


图 6-1　大数据的国外主流厂商

开源基础技术之上，主流的大数据厂商分为大数据分析基础设施、大数据操作基础设施、大数据基础设施云服务、传统结构化数据库、商业智能、可视化等领域。

1. 大数据分析基础设施

大数据分析基础设施主要是指 Hadoop 发行版产品，主要的厂商和产品有：Cloudera、Hortonworks、MapR，这是 3 家主要的 Hadoop 发行版产品的提供商，下一节会具体介绍其产品。其他主流厂商的产品包括 HP 的 Vertica、EMC 的 Greenplum HD、IBM 的 BigInsights 以及 ParAccel、InfoBright、Kognitio、Calpont、Exasol、Datastax 等。

2. 大数据操作基础设施

大数据操作基础设施主要是指企业级的 NoSQL 数据库和 SQL on Hadoop 产品，主要产品有 Couchbase、Hadapt、Teradata、10gen、Terracotta、MarkLogic、VoltDB 等。例如，Couchbase 和 MarkLogic 等都是企业级的商用 NoSQL 数据库。

3. 大数据基础设施云服务(IAAS)

基于大数据基础设施提供的云服务有 Amazon Web Services Elastic MapReduce、Google BigQuery、Infochimps、Microsoft Windows Azure 等。

4. 关系型数据库

关系型数据库产品 Oracle、Microsoft SQL Server、MySQL、PostgreSQL、MemSQL、SAP 收购的 Sybase、IBM DB2 等。

5. 数据云服务（DAAS）

DaaS 的主要产品有：Gnip、Datasift、Space Curve、Factual、Windows Azure Marketplace、LexisNexis、Loqate、Kaggle、 Knoema、Inrix 等。例如，Windows Azure Marketplace 就是基于 Windows Azure 云计算平台，供数据供应商和开发人员购买和销售数据集和应用程序的在线市场。

6. 商业智能产品

商业智能产品主要有：Oracle 的 Hyperion、SAP Business Objects，微软的 Microsoft Business Intelligence、IBM Cognos、SAS、MicroStrategy、GoodData，HP 的 Autonomy、QlikView、Chart.io、Domo、Bime、RJMetrics 等。这一类产品通常兼具分析和可视化的能力。

7. 分析和可视化应用

分析和可视化应用主要产品有：SAS、Teradata Aster、Tableau Software、Palantir、MetaMarkets、Visual.ly、KarmaSphere、EMC Greenplum、Platfora、ClearStory Data、Dataspora、Centrifuge、Cirro、Ayata、Alteryx、Datameer、Panopticon、Tibco、Opera、Metalayer、Pentaho。例如，EMC Greenplum 套件能对各种类型数据进行分析和可视化展现。Teradata 收购的 Aster Data 是高级分析和管理各种非结构化数据领域的重要厂商。

8. 日志应用

日志数据应用主要产品有：Splunk、Loggly、Sumo Logic。例如，Splunk 是一个可运行于各种平台的 IT 数据、日志分析软件。

9. 广告/媒体应用

广告/媒体应用主要产品有：Media Science、Bluefin Labs、CollectiveI、Recorded Future、LuckySort、DataXu、RocketFuel、Turn 等。例如，RocketFuel 是一家广告优化公司，Rocket Fuel 每天处理 15 亿次品牌广告展示，广告效果完全基于数据来进行改善。

10. 垂直应用

大数据垂直应用的主要产品有：Predictive Policing、BloomReach、Atigeo、Myrrix。例如，BloomReach 公司面向市场营销开发大数据应用（BDA），通过机器学习、网络爬虫和搜索技术来挖掘数据，对网站的数据进行分析，然后设法为网站带来更多的流量，从而给他们的客户带来更多的利润。

6.1.2　国内产业链主要力量

如图 6-2 所示为国内大数据的产业链现状。

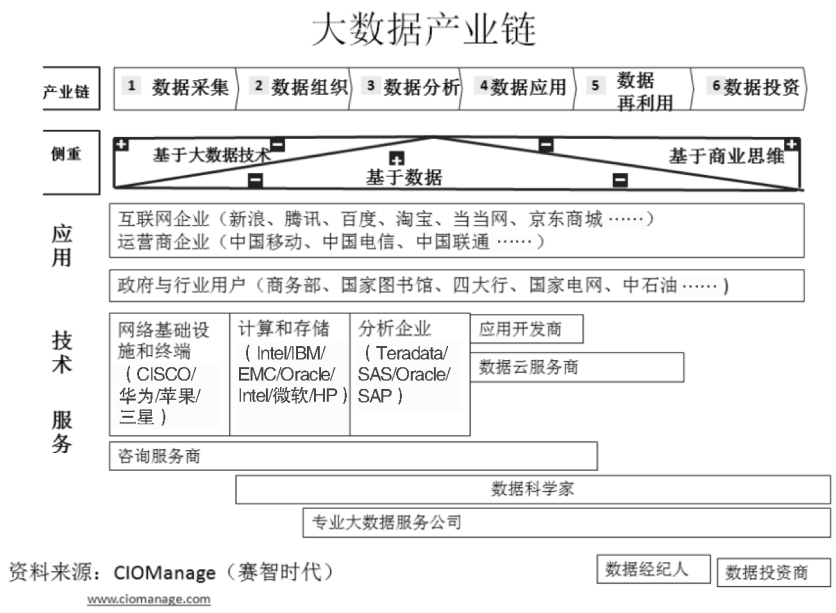


图 6-2　国内大数据的产业链现状

国内在互联网和行业应用领域的大数据应用也处于较好的水平，例如淘宝网、百度、腾讯、新浪等公司在大数据技术研发和应用方面均处于较高的水平。但在大数据产品和技术服务领域却落后于国际厂商，尽管这的确是一次国内企业迎头赶上的大好机会。

6.2 主流厂商解决方案

Hadoop 从 2008 年作为 Apache 开源项目发布以来，它就一直是大数据技术的核心，它结合了成本低、可扩展性佳以及无须构建预定义模型就能灵活地处理任何数据等诸多优点。Hadoop 取得的成功促使主流市场对其稳定性、成熟的管理等提出更高的需求。Hadoop 厂商们正在努力开发成熟可靠的商用发行版，这个社区中起始阶段就非常有影响力的主要厂商包括 Cloudera 和亚马逊。Cloudera 是开山鼻祖，现在也是 Hadoop 软件的最主要来源，它拥有 CDH 发行版和配套的管理软件，它也是为 Hadoop 提供企业支持和培训服务的最大供应商。亚马逊很早就进入了大数据领域，其亚马逊弹性 MapReduce 服务在公有云中运行 Hadoop。2011 年起，MapR 和 Hortonworks（后者从雅虎拆分出来）也备受瞩目，它们宣布了各自的 Hadoop 软件发行版，另外提供支持和培训服务。数据处理是一方面，大多数 Hadoop 用户最终希望实现的是分析数据。Datameer、Hadapt 和 Karmasphere 等专门针对 Hadoop 数据访问、商业智能和分析工具的厂商涌现出来，也让 SQL on Haddop 成为一种流行。Hadoop 迈向主流的标志是在 2011 年，它得到了 5 家主要的数据库和数据管理厂商的积极接受，EMC、IBM、微软和甲骨文都纷纷进入 Hadoop 领域展开竞争。IBM 和 EMC 发布了各自的 Hadoop 发行版，后者还与 MapR 结为合作伙伴。微软和甲骨文则分别与 Hortonworks 和 Cloudera 合作。IBM、EMC 和甲骨文都发布了专门定制的硬件设备，随时可以运行 Hadoop。

下面将重点分析 9 个主要厂商的应用解决方案，为读者选用商业化解决方案提供一些参考。本书仅就方案框架进行介绍，详细解决方案可以咨询厂商，关于厂商产品性能的比较可以参考基准测试和行业解决方案研究报告[2]。

6.2.1 Cloundera

作为 Hadoop 商用软件和服务提供商，Cloudera 公司自 2008 年以来就一直致力于将开

[2] 《2013 年大数据解决方案性价比研究报告》，CIOManage（赛智时代），2013 年。

源 Apache Hadoop 打造成一款供企业级用户使用的可靠平台。这家公司有 100 多个客户。Cloudera 企业解决方案包括 Hadoop 软件发行版、Cloudera 管理器及支持，近期标价为每年每个节点 4000 美元（不包括硬件），如图 6-3 所示为 Cloudera 的 Hadoop 套件，图 6-4 所示为 Cloudera 的产品和服务。

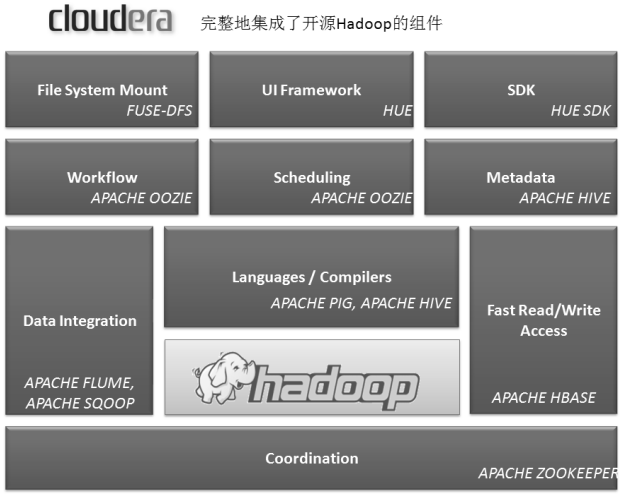


图 6-3　Cloudera 的 Hadoop 套件

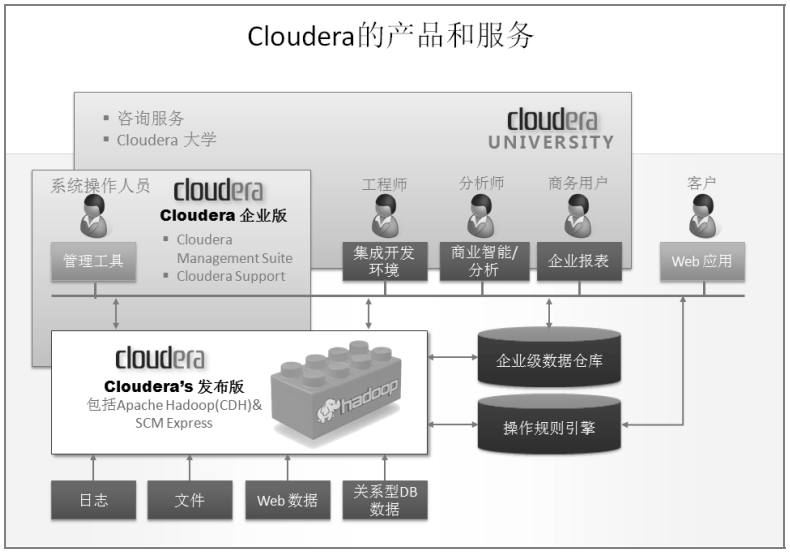


图 6-4　Cloudera 的产品和服务

6.2.2 Hortonworks

Hortonworks 公司在 2011 年从雅虎拆分出来，是完全致力于推进 Hadoop 开源平台的独立公司。这支雅虎团队开发了 Hadoop 平台背后的大部分代码，在引领该平台的开拓方面起到了很大的推动作用。

Hortonworks 推出了 Hortonworks 数据平台（HDP）。Hortonworks 还提供 Hadoop 支持、培训和咨询，与 Cloudera 和 MapR 加大了竞争力度。Hortonworks 与微软等很多主流 IT 厂商结为合作伙伴，例如通过与微软的合作关系，Hortonworks 将帮助微软开发与 Windows 兼容同时恪守 Apache 开源项目原则的 Hadoop 版本。如图 6-5 所示为 Hortonworks HDP 的生态系统。

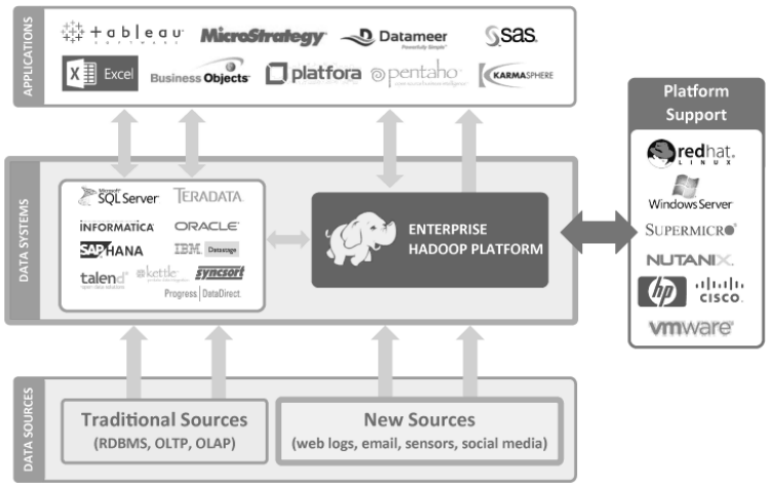


图 6-5　Hortonworks HDP 的生态系统

6.2.3 MapR

MapR 公司在 Hadoop 领域特立独行，它提供了一款独特的发行版。它从开源 Apache 项目获取了该公司所需的组件，同时摈弃了它不喜欢的组件（特别是 Hadoop 分布式文件系统，即 HDFS，MapR 认为这是单一故障点，并将它换成了基于 UNIX 的网络文件系统 NFS）。如图 6-6 所示为 MapR 的产品和服务。MapR 将其 M5 商业 Hadoop 发行版与支持、培训和咨询等服务（M3 发行版是免费的，还与 Apache Hadoop 百分之百兼容）结合起来。

MapR 与 EMC 结为了合作伙伴，EMC 采用 M5 作为其 EMC Greenplum HD 企业版的基础。

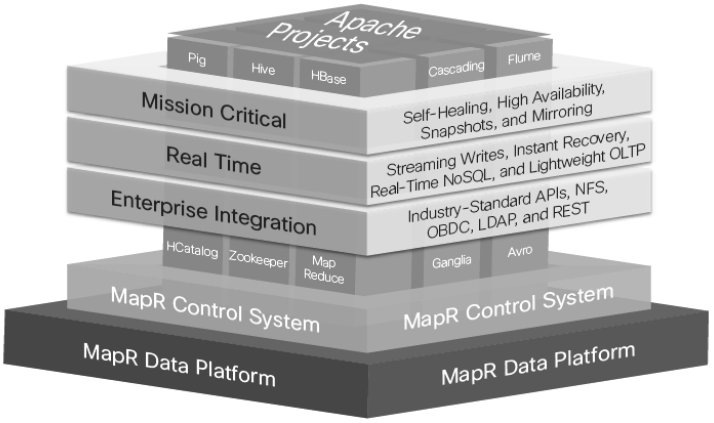


图 6-6　MapR 的产品和服务

6.2.4　IBM

IBM 的大数据平台是以 InfoSphere BigInsights 软件为核心[3]的，该软件包包括 Apache Hadoop 发行版、面向 MapReduce 编程的 Pig 编程语言、针对 IBM 的 DB2 数据库的连接件以及 IBM BigSheets。BigSheets 是基于浏览器的电子表格，可用于探究和分析 Hadoop 里面的数据。如图 6-7 所示为 IBM 的大数据平台。

IBM 对流计算提供 InfoSphere Stream 产品。InfoSphere Streams 是一个高性能的流计算平台，用来持续高速地分析海量数据流。它的主要特点包括：对事件和变化的要求做出快速回应，处理时延在几百微秒的水平；持续对海量数据流进行分析，处理能力达到百万 msg/s；快速适应数据格式和类型的变化，支持结构化和非结构化数据，支持多种数据源；为流数据处理提供可靠性、可扩展性和分布式的管理；为共享信息提供安全保障。 IBM 将 Streams 主要运用于企业级的大数据分析业务中，而随着互联网的兴起，许多互联网企业也对流计算越来越关注。

IBM 通过 InfoSphere 信息整合平台将 BigInsights 与传统信息管理、数据库、数据仓库等解决方案整合在一起，提供大数据整体解决方案。其中，InfoSphere DataStage 是一个实

[3] www.ibm.com.cn。

用的集成化的图形化设计工具，支持各种大数据作业的集成和协作，在整个 IBM 大数据解决方案中是一个非常实用的工具。

图 6-7　IBM 的大数据平台

IBM 提出的大数据分析一体化机产品是 PureData，它是 PureSystem 中的产品。PureData 包含 3 种类型。

- PureData System for Transactions 系统主要用于处理各种大数据的联机事务处理（OLTP）以及商业分析任务。例如，在交易处理方面，PureData 可以在单一系统整合多种业务数据库，优化大量交易处理。PureData 集成了基于 IBM DB2 数据库的 PureScale 集群技术。
- PureData System for Operational Analytics 系统主要用于商业分析应用，能够对数以千计的交易进行实时分析，例如金融交易中的欺诈监测和趋势发现。
- PureData System for Analytics 则基于 IBM 在 2010 年收购的 Netezza 数据仓库应用，该系统能够处理结构化和非结构化数据。

每台 PureData 售价近期约 50 万美元。

IBM 产品之间的数据集成标准和协议较为复杂。传统结构化数据库和数据仓库的数据

通过 JDBC 和 ODBC 连接器实现与 BI 系统 Cognos 的连接。BigInsights 则通过基于 JDBC 的 Hive 和基于 HTTP 的 REST API 来实现与 Cognos 的连接。传统数据库与 BigInsights 之间可以通过用户定义函数（UDF）来实现数据的装载。查询分析是 IBM 高级声明性查询语言 Jaql、Pig 语言和 Hive。文本分析则通过 BigSheets 以 CSV 格式传给 Cognos Insigh，如图 6-8 所示。

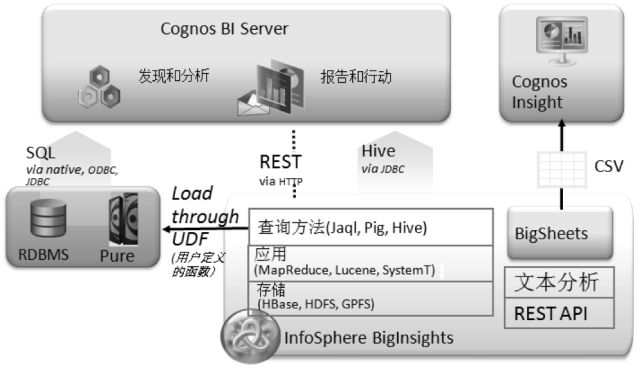


图 6-8　IBM Bigsights 的系统整合

IBM 通过其智慧云企业（SmartCloud Enterprise）的基础架构，将 BigInsights 和 BigSheets 作为一项服务来提供。这项服务分基础版和企业版，客户不必购买支持性硬件，也不需要 IT 专门知识，就可以学习和试用大数据处理和分析功能。IBM 声称，客户用不了 30 分钟就能搭建起 Hadoop 集群，并将数据转移到集群里面，近期数据处理费用是每个集群每小时 60 美分起价。

6.2.5　Oracle

甲骨文大数据机（Oracle Big Data Appliance）将甲骨文—Sun 的分布式计算平台与 Cloudera 的 Apache Hadoop 发行版、Cloudera 管理器管理控制台、R 分析软件的开源发行版以及甲骨文 NoSQL 数据库组合起来，如图 6-9 所示，一体化提供给客户使用。甲骨文解决方案还包括连接件，让数据能够在大数据机与甲骨文内存计算平台 Exadata 或传统的甲骨文数据库部署环境之间来回传送，如图 6-10 所示。甲骨文为这套综合的软硬件“工程一体化系统”提供了一线支持。Oracle NoSQL 数据库是大数据机的一个关键部分，是基于 Berkeley DB 的，面向海量非标准数据的用户，Oracle 将提供免费的社区版和收费的商业版本。

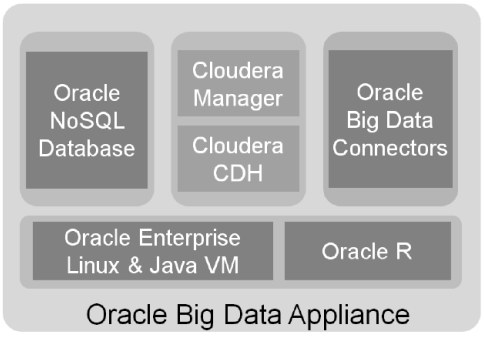


图 6-9　Oracle 大数据机的模块

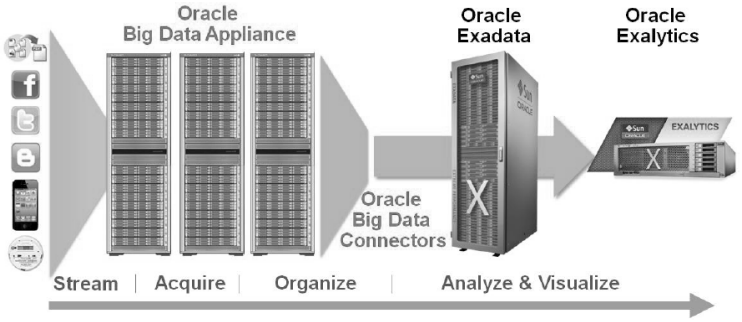


图 6-10　Oracle 的大数据解决方案

客户可以配置和使用大数据机捆绑软件。可能是完全的 Hadoop、完全的 NoSQL 或者在同一平台上两种节点对半分。该设备将完全通过全机架（full-rack）配置来提供，每个机架配备 864GB 主内存、216 个处理器核心、648TB 原始磁盘存储容量，以及节点之间每秒 40 千兆的 InifiniBand 内部连接。近期，软硬件总计售价将达到 45 万美元，每年还要收取 12%的软硬件支持费。

6.2.6　EMC

EMC 大数据解决方案的核心是 Greenplum 统一分析平台（UAP），数据团队和分析团队可以在该平台上无缝地共享信息、协作分析。UAP 包括 ECM Greenplum 关系型数据库、EMC Greenplum HD Hadoop 发行版和 EMC Greenplum Chorus，Chorus 是一种协作式、类似社交网络的界面，可供数据分析团队进行协作。UAP 软件将数据访问、管理和工作流统

一起来，并与其他数据源和数据处理方法相互关联。无论团队成员是数据科学家、数据集成专家和商业智能分析员，还是数据库管理员和业务部门的用户及管理人员。

EMC 的大数据机是模块化的 EMC 数据计算设备（DCA），它能够在一个设备里面运行并扩展 Greenplum 关系型数据库和 Greenplum HD 节点，如图 6-11 所示。DCA 提供了一个共享的指挥中心（Command Center）界面，让管理员可以监控、管理和配置 Greenplum 数据库和 Hadoop 系统性能及容量。

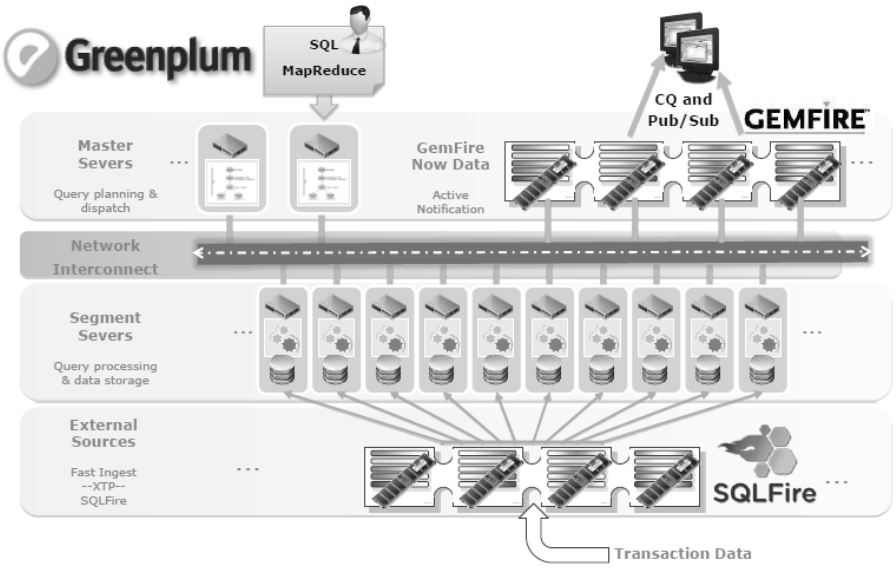


图 6-11　EMC 的大数据平台

6.2.7　Intel

Intel 的企业级 Hadoop 版的发行版是基于开源的 Apache 架构来构建的，如图 6-12 所示为 Intel 企业级大数据应用整体架构。Intel 的传统优势在底层 CPU 技术，毕竟 X86 架构的服务器是 Hadoop 集群常用配置，因此在基础技术层面的优化将提升 Intel 大数据平台的竞争能力。例如，在内存计算、数据压缩和数据保护等方面的优化，以帮助处理庞大的数据集；在高带宽、低延迟和多核处理能力方面的优化，以改进吞吐率和速度，加速大数据集的分析；内置于硬件中的安全性，以有效保护分布式基础设施并加快数据加密速度；在 SATA 接口的固态硬盘（SSD）方面的优化，以为原始存储提供高性能、高吞吐率的支持。

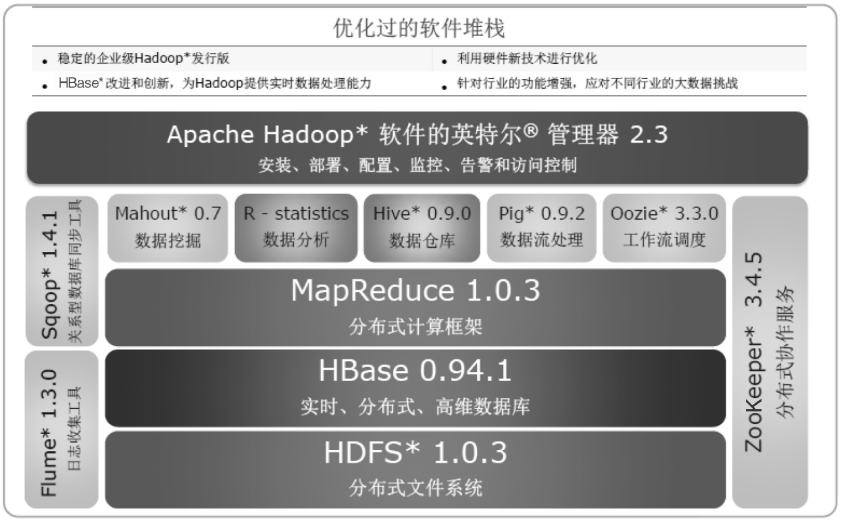


图 6-12　Intel 企业级大数据应用整体架构

Intel 是在中国市场上最为积极的大数据平台推动者，Intel IT 中心经常有很多优秀的应用经验提供给业界。Intel 提供了大数据应用指南和工具，如规划指南、同行调研、使用案例、行业分析洞察以及现场活动，来提供大数据应用的指导，帮助推动大数据应用，并提供 Apache Hadoop 的 Intel 云构建计划（Cloud Builders）参考架构等。

作为开放数据中心联盟（ODCA）的技术顾问，Intel 积极参与数据服务工作组（Data Services Workgroup）的工作，帮助其制定使用模型的需求，以支持安全地收集、管理和分析大数据；也推动了基准测试套件的发展（如针对 Hadoop 的 HiBench）；以及制定互操作性标准。

6.2.8　SAP

SAP HANA 内存计算是 SAP 的大数据解决方案。HANA 集成了 SAP 在内存计算和数据库方面的诸多组件，内存计算数据库（IMDB）、Sybase 的 Replication 技术，以及 SAP STR（Landscape Transformation Replicator）等。SAP IMDB 是一个混合式的内存数据库，包含行存储、列存储和基于对象存储的数据库技术，如此设计的主要目的是用来充分挖掘和使用现代多核 CPU 架构设计所带来的并发处理能力。内存计算整合了列式数据库的技术和内存计算的技术，无须索引，无须物化视图，无须调优，性能上更好，如图 6-13 所示。

在存储结构上，HANA 分为内存、闪存（持久层）和磁盘存储。HANA 基于内存数据库，在内存中进行实时数据同步操作或者实时数据的更新是很快的，但是磁盘的读写速度往往和内存的速度有差异。为了解决这个问题，在硬件层面，HANA 提供了一个闪存用来同步保存内存数据库中的日志信息（闪存有点像快速缓存，即使断电，还有数据保留，这个闪存有 2～4TB 左右），并且生成保存点，生成持久层。从 HANA 的内存写到持久层的数据，包含了两个部分：数据和日志。这个过程是持续不断的过程，当然中间有一定的时间间隔，其实持久层就是 HANA 内存数据库某个时点的一个完整的镜像备份，以及这个备份之后所有发生的数据库更新日志信息（在停电前成功执行完毕的）。磁盘存储用于保存和备份 HANA 的数据库，因为持久层的容积是有限的，所以 HANA 的备份都放在外部的物理存储上，比如高速率的硬盘或者其他设备。在备份数据和恢复数据的时候会用到，比如重启服务器。

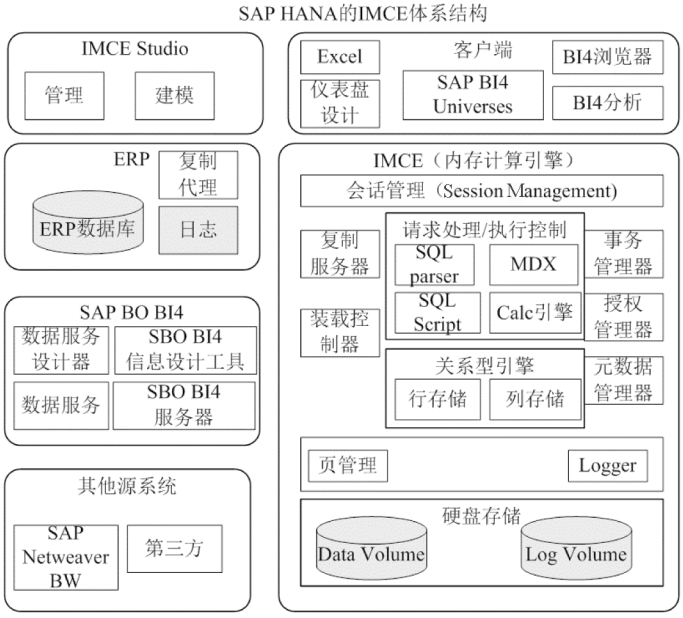


图 6-13　SAP HANA 的 IMCE（内存计算引擎）

SAP HANA 在内存中进行计算，它使用了 3 种复制技术，把数据库中的数据复制到内存中：一是基于触发器的复制，它实时捕捉 SAP ERP 的数据库系统的修改变化，然后几乎是实时地同步到 HANA 中；二是基于 ETL 的复制，它通过 ETL 技术把数据装载到 HANA 中；三是基于日志的复制，它根据日志文件来把数据库复制到 HANA，如图 3-32 所示。

6.2.9 Teradata

Teradata 是一家数据仓库厂商，擅长传统数据仓库和数据挖掘分析工作。Aster 原来是一家专注于 MPP 的厂商，并较早启动 SQL-MapReduce 模块研发，成为较为成功的大数据产品。Teradata 收购 Aster 后，把 Teradata Aster 作为其大数据平台的核心产品，Teradata Aster 平台如图 6-14 所示。

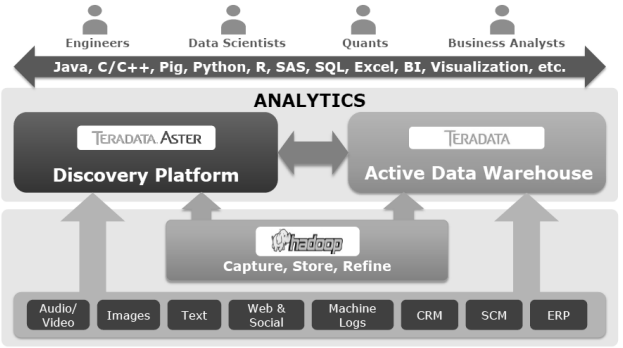


图 6-14　Teradata Aster 平台

Teradata Aster 采用的是 Hortonworks Hadoop 发行版。利用 MapReduce 将多结构数据在数据库内分析处理，并提供标准 SQL 和基于 MapReduce 的预封装模块，以及数据系统之间高速数据传输的 Teradata-Aster Adaptor 连接器。

Teradata Aster 的主要模块如下。

- Aster SQL-H 软件，它使用 Hadoop 的全新元数据目录 Apache Hcatolog，能通过标准商业智能应用来实时访问储存在 Apache Hadoop 内的大数据集。
- Viewpoint Portlets，它是基于 Web 的独立的管理和监控应用套件，支持 Aster 数据库和 Apache Hadoop 管理员访问重要系统的管理信息。
- Aster 管理控制台（AMC），它用于配置、管理和监控数据、应用程序和基础设施，它通过直观的图形界面，支持仪表盘汇总监控、查询和查看进程执行视图，以及向下钻取（drill-down）查询。
- Teradata Server Management，它提供服务器管理功能来主动监控系统。
- 海量并行处理，通过嵌入式 SQL 和 MapReduce 引擎提供数据分析的并行处理能力。

第 7 章

大数据应用的未来挑战和趋势

本章是全书的结尾，阐述大数据应用在隐私保护、技术标准和数据治理方面的未来挑战和大数据时代的商用社会发展趋势。

7.1 隐私保护

大数据应用产生了很大的数据分析价值，这种价值会随着信息的公开和共享进一步放大。但大数据的公开和隐私保护是一把双刃剑，享用信息公开价值的同时也带来了公民隐私保护的重大挑战。由于大数据技术确实存在记录人们生活信息的可能性，随着它的普及可能会侵犯到人们的隐私。例如，一些商业机构为了商业目的而私下收集和应用个人数据，可能很大程度侵犯了个人的隐私。其实，即使各类机构以公开的、个人允许的方式收集了个人数据，如果这些数据被用于其他场合的分析和应用，也可能侵犯个人隐私。就个人隐私而言，不同时期零散地公开的信息和经过分析挖掘后一次性完整公开的信息，即使内容相同，也是有本质的区别的。侵犯隐私的问题至关重要，如果解决不好，它很有可能使大数据的应用受到重大影响。

《大数据时代：生活、工作与思维的大变革》作者维克托·迈尔·舍恩伯格在他的第 2 部著作《删除：大数据取舍之道》一书中提出了对策[1]，他提出人们应秉承数字化节制，保护信息隐私权，建设数字隐私权基础设施，并调整人类的现有认知，打造良性的信息生态和实现完全的语境化。所谓“数字化节制”指的是人作为信息社会的主体，要积极发挥能动性，既要拥抱大数据，也要审慎控制过多的个人信息对外发布。当然，这是一种艰难的权衡，积极分享个人信息毫无疑问可以带来诸多便利，相反控制信息分享会减少一部分便利。保护信息隐私权、建设数字隐私权基础设施、打造良性的信息生态，需要政府及企业、法学界等方面加强协商，从政策、技术、法律等方面采取措施，保障大数据应用和隐私保护。信息隐私权最基本的形式是给予个人选择是否共享信息的权利，严禁任何其他个人或组织在法律许可和信息所有者许可之外，滥用个人信息。尽管对“信息隐私权”这个概念的界定，必然将引起较大争议，但唯有通过广泛讨论，方能为共识的取得创造可能。在此基础上，政府应建设数字隐私权基础设施，并推动相关的立法进程。

7.1.1 法律保护

从大数据安全和隐私保护的法律层面，应启动大数据立法，解决数据隐私保护、数据主权归属问题，并规定相应的法律责任。既要为依法合理地收集处理大数据和公开信息提供保障，也要确保数据处理过程中个人隐私不被泄露，不被用于服务和统计以外的目的。法律需要针对两方面情况做出相应的规定，既要保障大数据合法地开发利用，也要明确指出在此过程中，个人和企业信息既不得向统计等部门以外的第三方提供，也不得用来对个人和企业进行处罚，更不得对社会发布。法律要细化处罚条款。一旦发生上述情况，相关违法人员必须承担法律责任。另一种比较特殊的情况则是，为了保卫国家安全或查处犯罪行为，需要采取一些特殊手段。但这也必须在法律的框架内进行。在证据和理由充分的情况下，仅针对特定的目标，并经过必要的法律程序。总之，需要尽快立法以适应大数据时代保护公民隐私的需要。

世界各国对数字资产相关的个人隐私保护法律的侧重点不同。在美国，宪法中没有清楚的隐私权的条款。如果把隐私权加入宪法意味着美国权利和自由法案的大量附带条件的修改。对于个人隐私保护的基本精神是借鉴对私人财产保护的角度来进行探讨的。同时，个人数据的公开表现出宪法对言论自由的保护精神。所以，美国政府站在促进商业活动的

[1] 《删除：大数据取舍之道》，维克托·迈尔·舍恩伯格著，袁杰译，浙江人民出版社，2013 年 1 月。

立场，认为个人也必须适当公开一些信息，才有助于市场竞争及交易。白宫和私人部门相信自我规范和自律是足够的，因此没有新的法律被需要（唯一例外是医学记录）。当然，关于个人信息的专门分类在美国联邦法律里进行了补充规范，如财务报告、信用报告、视频出租等方面的要求，以及没有法律保护的领域，如在互联网上的个人隐私保护。对于隐私保护，美国更强调通过技术进步及相关保护程序来增强隐私权的保护。

尽管如此，美国的个人隐私保护法律体系也是全世界最健全的[2]，包括如下内容。

- 1974 年的《美国隐私法案》，保护被联邦政府的执行机构收集的数据的隐私。
- 1984 年的《美国计算机欺诈和滥用法案》，被偷窃的价值是否大于 10 万美金来决定 1～20 年的处罚年限。
- 1986 年的《美国电子通信隐私法案》，反对窃听，例外是法院指令和 ISP。
- 1996 年的《美国经济间谍法案》。
- 1996 年的 HIPAA，保护个人医学记录的隐私。
- 1999 年的《Gramm-Leach-Bliley Act》，保护金融机构消费者数据的隐私。
- 2001 年的《USA Patriot Act》、《美国电子基金传输法案》、《美国信息自由法案》等。

欧盟的个人隐私保护则很严厉，也更加强调政府在其中的重要作用。2004 年，欧盟采用了《隐私电子通信的指引》（Privacy Electronic Communications Directive），禁止没有被告知同意的情况下进行数据的二次使用，在非欧盟国家之间禁止数据的传递，除非有足够的隐私保护。之前，早在 1994 年和 1998 年，欧盟就两次颁布了《EU 数据保护法案》，数据隐私保护远强于美国。

我国个人信息隐私保护还处于起步阶段。首个个人信息保护国家标准《信息安全技术公共及商用服务信息系统个人信息保护指南》于 2013 年 2 月 1 日起正式实施，该标准对个人信息的使用和处理做出明确规定，保护公民对个人信息处理的知情权和决定权。

建立以个人隐私保护的法律基础，将为依法应用大数据保驾护航。

[2] http://www.cs.purdue.edu/people/bb，”Introduction to Privacy in Computing”，Bharat Bhargava。

7.1.2 技术保护

在大数据隐私保护的技术层面，要防止不法分子侵入系统，盗取个人隐私信息；要限制个人信息掌握者的权限，使每个层级的相关人员只能掌握相应的有限信息；要对不信任的外部合作伙伴进行外包分析，防止数据窃取；要防范通过数据共享的无意泄露等。隐私保护技术（Privacy-Enhancing Technologies，缩写为 PETs）主要分为 3 类：一是保护使用者的身份的技术，二是保护被使用数据对象的身份的技术，三是保护个人数据的秘密性和完整性的技术。

1. 保护用户身份

保护用户身份的技术主要包括如下内容。

- 匿名：用户使用资源和服务时，不公开他的身份。
- 使用假名：用户不公开他的身份，使用假名来使用资源和服务。
- 不可见：用户使用资源和服务时，没有其他人能看到资源和服务正在被使用。
- 不能被链接：在彼此通信过程中，发送者和接收者不能被识别出。

2. 保护被使用对象的身份

主要技术是数据主题的去个性化（匿名化）技术。

- 完全的去个性化：数据被匿名后处理，以至于数据主题不再能被认出。
- 实用的去个性化：个人数据被修改，以至于关于个人的或物理环境的信息不再被识别，或者只能通过不成比例的时间、成本和劳力代价才能被识别，或者只被能识别的人所识别。
- 去个性化的控制包括：统计数据库的推理的控制、数据挖掘的隐私保护方法、大数据的隐私保护方法。

3. 保护个人数据的秘密性和完整性

- 增强隐私保护身份管理。
- 受限的访问控制。

- 企业隐私政策。
- 隐写术，即信息隐藏方法。信息隐藏指的是不让除预期的接收者之外的任何人知晓信息的传递事件或者信息的内容。
- 专门的工具，例如 P3P（Platform for Privacy Preferences，个人隐私倾向平台）。P3P 是由全球资讯联盟网所开发的一个标准，能够保护在线隐私权，在浏览网页时可以选择网名是否被第三方收集并利用自己的个人信息。如果一个站点不遵守 P3P 标准的话，那么有关它的 Cookies 将被自动拒绝，并且 P3P 还能够自动识破多种 Cookies 的嵌入方式。

近两年，世界经济论坛发表了两份关于大数据隐私的报告[3、4]，提出了建议性的解决方案。该解决方案是通过高端科技来保护隐私。这份报告是由众多研究隐私保护人士和组织完成的，由世界经济论坛、论坛成员国的政府官员、隐私支持者和商业高管们赞助。其中 2013 年的报告题目为“解除个人信息的价值：从收集到使用”，该报告建议将重心从管理转移到限制使用数据上来。根据此份报告，通过新型技术手段以限制使用个人数据能够让个人控制自己的信息安全，同时还能够使重要的数据资产得以自由利用。这份报告也回应了早期的隐私保护法案。于 1970 年通过的《公平信用报告法》（The Fair Credit Reporting Act）就是对大型主机侵犯人们隐私的有效防护，该法案允许信用咨询公司收集个人财务信息，但收集所得信息只能用在 3 个方面：信用、保险以及就业。此次世界经济论坛推出的这份报告建议在未来，所有数据的收集都应该通过密码。所有对于数据的使用都应该登记，同时对于那些违反规定的人要采取处罚措施。例如，智能手机应用存储了其并不需要的多余数据，很有可能是对于数据的违规操作。

7.1.3　理念革新

在大数据隐私保护的理念层面，要进行意识和理念的革新。加大大数据分析和共享的力度，但要建立信任机制，并变革责任体系。

[3] http://www3.weforum.org/docs/WEF_IT_UnlockingValuePersonalData_CollectionUsage_Report_2013.pdf。

[4] http://www3.weforum.org/docs/WEF_IT_RethinkingPersonalData_Report_2012.pdf。

1. 信任为基础的隐私交流

信任和隐私保护是紧密相关的。如果能够建立信任，那么实体之间是能够进行隐私的交流的。基于信任的隐私交流理念和机制期望能够加快大数据应用并加强隐私保护。

2. 数据使用者承担责任

大数据时代，隐私规范的核心法则需要进行革新。一直以来，人们自主决定是否、如何以及经由谁来处理他们的信息，把这种控制权放在自己手中，是隐私规范的核心法则。但大数据的价值体现在潜在价值挖掘和数据的二次利用，但在收集数据时尚未考虑或者意识到数据如何被二次利用，因此“告知与许可”就很难操作。维克托·迈尔·舍恩伯格认为大数据应该创造不一样的隐私保护模式，更着重于数据使用者为其行为承担责任，而不是收集数据之初取得个人同意。因此，数据使用者的责任承担体系建设就更为重要了。

7.2 技术标准

大数据平台和工具层出不穷，标准各异，增加了大数据的应用难度，并且缺乏标准也使得技术的发展缺乏持续性，使得一些技术昙花一现。因此，建立大数据技术标准是实现大数据大规模应用的基础性工作，也是推进大数据评价、测试和产业化的重要工作。

7.2.1 ISO 标准化进展

在 ISO 中，大数据和分析相关的技术委员会包括如下几个。

- JTC1/SC 32 Data management and interchange，数据管理和交换标准。
- JTC1/SC 06 Telecommunications and information exchange，通信和信息交换标准。
- JTC1/SC 39 Sustainability for and by Information Technology，信息技术的可持续性。

JTC1/SC 32 技术委员会是专门从事数据管理和交换标准化工作的，包含 4 个委员会，分别是 WG1 e-Business（电子商务）、WG2 Metadata（元数据）、WG3 Database Languages（数据库语言）、WG4 SQL Multimedia & Application Packages（SQL 多媒体和应用包）。例如，JTC1/SC 32 在 ISO/IEC 19763 系列标准，建立了互操作的元模型框架规格手册；在

ISO/IEC 11799、20943、20944 等标准中定义了元数据登记的相关规范；在 ISO/IEC 9075 系列中定义了 SQL 数据库语言标准等。

SC32 WG4 是关于非结构复杂数据标准的工作组，其在 ISO/IEC 13249 标准中定义了用 SQL 对全文本、空间数据、图像数据、统计数据挖掘、复杂统计数据挖掘等进行操作的相关标准。2012 年 7 月，我国的非结构化数据管理标准工作组成立。北京航空航天大学、清华大学、浙江大学、中国人民大学、北京大学、中国科学院软件研究所等高校及研究机构，百度、用友、阿里云、拓尔思、中软等业界厂商发起成立了非结构化数据管理标准工作组，负责制定和完善我国非结构化数据管理领域的标准体系、制定我国非结构化数据管理相关国家标准并对口 ISO/IEC JTC1 SC32/WG4，参与非结构化数据管理的国际标准化工作。

当前，JTC1/SC 32 已经启动大数据和下一代分析的标准研究工作[5]，概念模型如图 7-1 所示，我国的标准化组织也加入了这部分工作。今后，我国在大数据、下一代分析学、Web 协作、社交网络等新领域标准化的研究和布局将逐步推进。

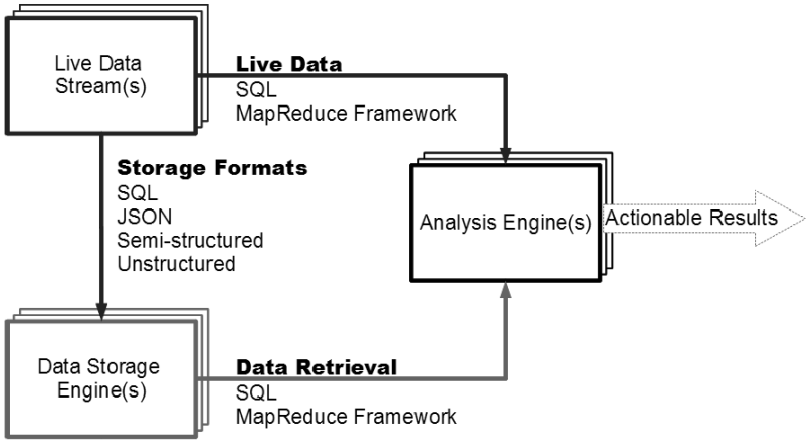


图 7-1　JTC1/SC 32 大数据标准研究的概念模型

7.2.2　评价基准和基准测试

在计算机和数据库领域赫赫有名的 TCP 基准测试结果是 IBM、HP、Oracle 等厂商进

[5] http://www.jtc1sc32.org/doc/N2351-2400/32N2388b-report_SG_big_data_analytics.pdf。

行产品推荐时经常要拿给客户看的性能指标，客户也往往把这些指标作为一个基准来进行参考。大数据应用也需要这样的基准测试。当前，大数据基准测试（或称为标杆比较）已经着手建立：一方面是 TPC 的基准测试向大数据领域进行推广，如 TPC-DS（TPC 关于复杂决策支持的基准）、TPC-H（TPC 针对特定查询的基准）等；另一方面则是大数据基准“另起炉灶”。大数据基准委员会[6]就试图建立一个端对端的应用层基准来评测各个大数据应用的性能，帮助更多企业按照基准的规范来应用大数据。大数据基准测试将成为比较各个主流的大数据硬件和软件技术、产品有效性的行业标准。关于大数据基准测试的研讨会已经进行了几次，这是由美国国家自然科学基金会支持（NSF）的，由 40 多家主流大数据企业参与的研讨会。讨论主题如表 7-1[7]所示。

表 7-1　大数据基准的关注点

关　注　点	可能的答案
谁关注基准	- 客户：客户需要通过基准测试来了解谁的产品功能合适，性能更优？因此，工作负载应很好地用文字、声明语言来表达，但不合适用程序语言来表达。 - 厂商：根据第三方对自身产品的基准测试来证明自己的产品是性能优秀的。 - 计算机科学/硬件研究机构：基准所涉及的专业领域佼佼者和新技术将从学术领域涌现出来，关于特定的基准将有利于培训学生
什么样的应用用于基准评测	- 一个某些用户能义务提供的应用。 - 一个基于经验数据的应用，例如在科学应用中的例子，可以很好佐证基准测试效果。 - 一个多渠道零售商的应用，例如为大数据应用进行改进的 TPC-DS 基准所采用的应用，特点是有成熟的模型（schema）、大规模数据生成器、执行规则和审计流程。 - 互联网规模应用的模拟，如在 Facebook 网站综合数据的管理
一个统一的基准还是多个基准	用一个基准来覆盖所有应用确实是可能的。但是，如果在这些被评价对象之间没有任何协同，例如在数据模型层面的协同，那么一个统一基准的效果并不好。所以，评价对象之间的协同很关键。例如，协同的 Facebook 应用可能可以提供一个单一基准的环境，用于评价点击流、数据存储/索引、Web 日志处理、图的遍历、图像和视频数据等
部件基准还是端对端应用基准	- 基准应包括独立部件的基准，并最终能将这些部件组合成为端到端应用的基准。 - 基准评价应至少包含一个抽取大数据的部件的基准评价，它具有共性，很多数据科学应用都有抽取大数据并可视化输出结果的功能，把数据推送到数据管理系统是一个基本需求

[6] http://clds.ucsd.edu/bdbc/community。

[7] http://www.snia.org/sites/default/files2/ABDS2012/Track1/ChaitanBaru_Setting_the_Direction_Big_Data.pdf。

续表

关　注　点	可能的答案
纸面规范驱动还是实施驱动	以实施来启动并同时开发规范。一些活动已经启动起来，如数据生成、排序和其他一些处理
数据从哪里来	- 下载数据不应是一个选择。 - 数据需要被快速生成。 - 从科学应用中来的实际数据集的样例是很好的选择，如 LSST（大型气象观测项目）的观测数据，仿真的输出结果等。 - 用已有的数据生成器（TPC-DS、TPC-H）。 - 数据有很好的特征并具有足够的通用性，要优于那些专门的数据
基准能用于创新或者竞争吗	基准应既用于竞争，也在内部用于创新和提升。TPC-H 就是一个很好的基准模型，能够同时驱动竞争和创新
大数据能重用现有的基准吗	可以重用。但是必须考虑这些基准工具要做哪些调整工作：增加多少功能？是不是能支持基准数据的规模扩大？是不是只能用 SQL？ 因此调整是必须的，至今没有一个基准能不调整就用于大数据基准测试，包括如下工具。 - Statistical Workload Injector for Map Reduce (SWIM) - GridMix3，Hadoop 自带开源工具 - TPC-DS - YCSB++ - Terasort，最好作为端到端情景的一个部分
什么是好的大数据基准	- 自我实现规模扩大，如 TPC-C - 在不同规模因素间可进行比较 - 结果应在不同规模间进行比较 - 技术无关（如果这对于应用很有意义） - 运行简单
与其他基准相比有什么特点	TPC 基准通常运行在超载的系统上，做峰值基准测试，因此基准安装的系统比客户的安装有更多的硬件。大数据基准可能正好相反。可能需要运行在比客户安装更小的安装环境。需要确认仿真环境是不是能够线性推广到客户环境？是不是存在推导的拐点

HiBench[8]是 Intel 公司提供的一套完整的 Hadoop 基准测试集合，由中国团队进行维护，它可以进行微基准测试（单词记数、排序、Terasort）、网络搜索算法测试（Nutch 索引、PageRank）、机器学习算法测试（贝叶斯分类、K 均值聚类）、分析查询算法测试，如表 7-2 所示。

表 7-2　HiBench 基准测试[9、10]

类　别	工作负载	简　介
微性能指标评测	排序	该工作负载使用 Hadoop RandomTextWriter 生成数据，并对数据进行排序，代表数据从一种格式转换为另一种格式的实际 MapReduce 任务
	字数统计	统计输入数据中每个单词的出现次数，输入数据使用 Hadoop RandomTextWriter 生成，代表从大型数据集中提取少量感兴趣的数据的实际 MapReduce 任务
	TeraSort	Terasort 是由微软的 Jim Gray 创建的标准 Benchmark，输入数据由 Hadoop TeraGen 产生
	增强的 DFSIO（dfsioe）	通过产生大量同时执行读写请求的任务来测试 Hadoop 机群的 HDFS 吞吐量
Web 搜索基准	Apache Nutch 索引	大规模搜索引擎索引是 MapReduce 的一个重要应用，这个负载测试 Nutch（Apache 的一个开源搜索引擎）的索引子系统，使用自动生成的 Web 数据，Web 数据中的链接和单词符合 Zipfian 分布。其中，Crawler 子系统用于分析内部 Wikipedia 镜像，会生成 84GB 压缩数据（约 240 万网页）作为工作负载输入
	页面排名	这个负载包含一种在 Hadoop 上的 PageRank 算法实现，使用自动生成的 Web 数据，Web 数据中的链接符合 Zipfian 分布
机器学习	贝叶斯分类	大规模机器学习是 MapReduce 的一个重要应用，这个负载测试 Mahout 0.7（Apache 的一个开源机器学习库）中的 Naïve Bayesian 训练器，输入数据是自动生成的文档，文档中的单词符合 Zipfian 分布

[8] https://github.com/intel-hadoop/HiBench。

[9] http://www.intel.hk/content/dam/www/public/us/en/documents/guides/getting-started-with-hadoop-planning-guide.pdf。

[10] http://blog.jeoygin.org/2012/12/hadoop-benchmarks.html#more-1018。

续表

类　别	工作负载	简　介
	K-Means 聚类	大规模机器学习是 MapReduce 的一个重要应用，这个负载测试 Mahout 0.7 中的 K-means 聚类算法，输入数据集由基于均匀分布和高斯分布的 GenKMeansDataset 产生
分析查询	Hive Join	这个负载的开发基于 SIGMOD 09 的一篇论文“A Comparison of Approaches to Large-Scale Data Analysis”[11]和 HIVE-396，包含执行典型 OLAP 查询的 Hive 查询，使用自动生成的 Web 数据，Web 数据中的链接符合 Zipfian 分布。通过计算单个只读表中每个小组的总和，模拟关系型数据表的复杂分析工作
	Hive Aggregation	同上，通过计算合并两个不同的数据表来计算每个小组的平均值的总和，模拟关系型数据表的复杂分析工作

7.2.3　标准套件

五花八门的大数据技术工具和不断更新的版本经常让大数据应用开发人员感到困惑。因此，产生一整套标准化的大数据处理和分析软件（Hadoop 堆栈）有望让用户更容易开发大规模的大数据分析系统，正如开源 LAMP 堆栈在过去的 10 年间带来了一整批 Web 2.0 服务那样，Apache 以及商业化的 Cloudera、Hortonworks、MapR、Intel 等公司在推进 Hadoop 标准套件方面做了很多工作，逐步规范着大数据处理和分析软件标准套件的组成。

7.3　大数据治理

数据治理是一个老话题，但在大数据应用的情景下，数据治理的复杂度在提高，给大数据应用提出很多新的挑战。如何建立组织大数据治理框架？如何保证大数据分析的质量？谁该为基于大数据分析而做出的决策负责？如何提升大数据应用和投资的价值？如何防范大数据应用中的风险？如何保证大数据应用与法规的遵从？这些都是大数据治理需要回答的问题。

[11] http://database.cs.brown.edu/sigmod09/benchmarks-sigmod09.pdf。

7.3.1 数据治理框架

公司治理是股东通过公司治理结构来对经营管理层进行管控的制度安排，如董事会的安排、财务审计、绩效考核等。IT 治理是公司治理的一个有机组成部分，它包含了领导力、组织结构和流程等制度和机制，确保 IT 维系和扩展组织的战略和目标[12]。数据治理则是企业投资者对企业数据资源开发和利用的一种管控制度安排，它包括标准、政策和流程、组织结构、技术架构等制度和机制，确保数据资源的开发和利用符合组织的业务和 IT 战略。

数据治理主要关注数据质量管理、数据保护、信息安全与隐私管理、数据风险审计等流程。数据治理的制度安排涉及标准、政策和流程、组织、技术等几个方面，如图 7-2 所示。在大数据时代，数据治理的范畴更广、内容也更多，但仍遵循基本的数据治理框架。大数据标准、技术等内容在前面的章节已经阐述，这里重点讨论大数据的质量管理和组织管理。

数据治理框架

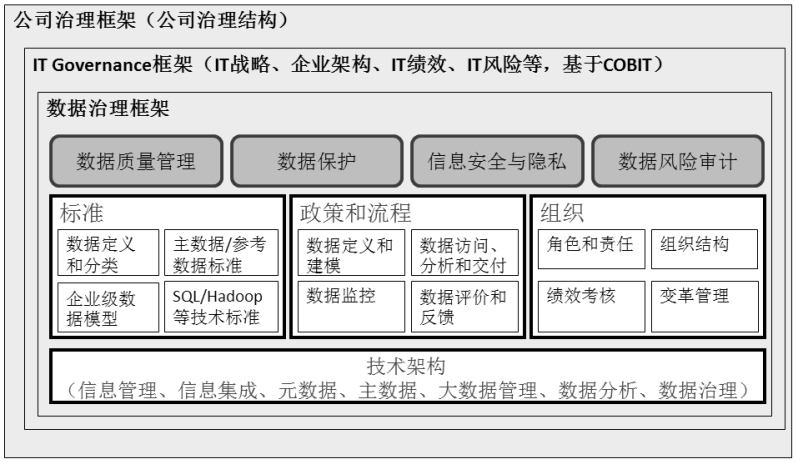


图 7-2 数据治理框架

7.3.2 数据质量管理

很多人认为，大数据的 4V 特性应该拓展到 5V，有必要加入第 5 个 V，即 Veracity，它关注数据的准确性。大体量的数据意味着传统的数据质量算法不经过优化性能很难满足

[12] 《IT 管理体系：战略、管理和服务》，赵刚、罗文著，电子工业出版社，2009 年。

大数据计算的要求；传统的数据质量算法基本上是基于关系数据的，对于非结构化文档类型支持不够；高速度也是一个严峻的挑战，数据采集过程本来就是数据质量问题的一个主要来源，采集速度快，又不能及时进行数据质量处理，会导致数据质量问题的堆积。因此，大数据质量管理面临很多新的挑战！

数据质量不仅仅是数据准确性的问题，它是一个综合性的指标要求，包含很多其他的重要内容，涉及很多定量和定性的指标，对数据质量的关注度不同所采取的数据质量管理措施也不同，如表 7-3 所示。大数据质量是大数据研究的一个热点领域，新技术和新方法会不断涌现。以现有方法和技术为基础，在大数据处理后进行质量的评价和管理，是一种过渡性的方案。

表 7-3　数据质量的内涵

指　标	关 注 点	措　施
数据准确性	数据是不是准确表达了现实或者一个能被证实的来源	- 为了提高数据的准确性，采用数据平衡计分卡、仪表盘等监控和纠正数据质量问题。 - 进行数据分类（Profiling），不同的级别采取不同的质量策略。 - 数据清洗、匹配和去重，提供优质数据。 - 为了保证数据的一致性，采用元数据管理和主数据管理，建立数据谱系、数据字典和业务术语表。 - 为了保证数据的完备性，采取复制、备份、消重等措施进行数据保护。 - 为了保证数据的可访问性，对结构化、非结构化数据提供高效的访问能力，集成对历史数据、主机数据、数据库数据、互联网数据等访问能力。 - 为了保证数据的及时性，采取高效的分布式计算、流计算、内存计算来实现高效的处理。 - 为了提高数据的可用性，建立分布式的容错机制、数据联盟或者实时的分区，提高数据可靠性。 - 为了提高数据完整性、完备性等，加强数据信息安全防范，采取加密、访问权控制等措施
数据完整性	在应当被相互关联的数据之间是否存在被破坏的连接	
数据一致性	有一个数据的个别的表达吗	
数据完备性	有任何关键信息的丢失吗	
数据唯一性	数据是价值唯一吗？即没有复制价值或记录	
数据可访问性	数据易于访问、可理解和被一致地使用吗	
数据精确性	数据是被以业务所需要的精确度来存储的吗	
数据及时性	信息的更新频率是足够满足业务需求的吗	
数据相关性	为了获得一个真正业务的表达，每一部分被存储的信息是重要的	
数据可使用性	被存储的信息在组织中是能用的	
数据的有用性	被存储的信息是能够应用于组织的	
数据的可信性	数据被认为是真实的、可信的程度	
数据的明确性	数据的每一部分有一个唯一的意义，并且容易被理解	
数据的客观性	数据是客观的、无偏见的、公正的，它不依赖于人的判断、解释或者评价	

7.3.3 大数据的组织、角色和责任

企业的大数据应用需要组织结构的支撑，需要对组织进行适应性的变革，以承担大数据的使命和责任。大数据应用的组织设计的基本流程是设定目标，分析环境与评估现状，设定业务愿景和方向，依据业务方向进行组织结构设计，并不断根据业务环境的变化实施组织变革，如图 7-3 所示。

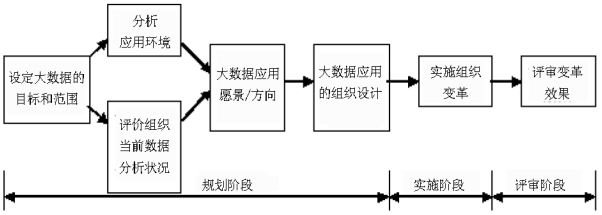


图 7-3 大数据应用的组织设计

目前，很多企业的大数据应用仍然主要由传统业务部门来推动，大数据作为一种技术手段用于改进分析和决策，信息技术部门作为技术的支撑方。但这种状况正在发生改变。在很多大数据应用的领先企业，大数据应用的部门从信息中心独立出来，成为了重要的业务创新部门。在他们看来，大数据应用是企业业务的重大创新，大数据应用组织未来将是企业重要的业务战略单元，因此大数据业务部门成为了战略性的业务事业部。这种类型的大数据事业部通常有首席信息官、数据科学家、数据分析师以及产品、营销和服务人员等角色。大数据应用事业部需要企业重要职能部门的重要支持，主要是来自信息技术部门的支持，毕竟大数据应用是一个与信息技术高度关联的业务。而两个部门的联动是通过首席信息官的统一协调来实现的，如图 7-4 所示。

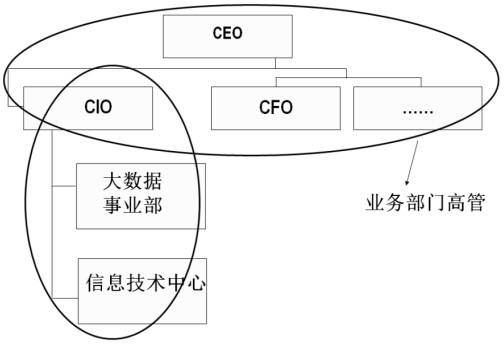


图 7-4 大数据应用的组织结构变化

组织结构确立后，岗位、职责和责任的明确就更为重要，如表 7-4 所示。

表 7-4　大数据应用的岗位和职责

职　位	职　责	责　任
首席信息官	管理企业的数据与信息资源的开发和利用，以及相关信息技术管理	对大数据应用负领导责任
数据科学家	从事大数据业务价值挖掘和商业化服务的研究和开发	对大数据应用的创新和研发负责
数据分析师	从事企业大数据分析和服务	对大数据业务分析结果负责
大数据技术架构师	从事企业大数据技术架构的设计和构建	对大数据技术架构负责
大数据管理员	从事企业大数据中的数据管理和数据治理	对大数据的质量管理、数据保护、数据安全等负责
大数据技术工程师	从事大数据技术系统的研发和运行维护	对大数据系统正常使用负责

7.4　适应商业社会的未来趋势

商业社会正在进入一个数据资源异常丰富的大数据时代！这对未来大数据的应用方向将产生重大影响。

7.4.1　从产品推销向数据营销的转变

在大数据时代，谁拥有数据，谁就拥有客户。商家与客户的关系不再是简单的卖方和买方的关系，依靠大数据应用，商家可能成为客户最信赖的朋友。他们了解客户的爱好，熟悉客户的朋友圈，能够给客户更多中肯的建议。他们不再只是生硬地向客户推销产品，而是把与客户和产品相关的关键数据、信息和知识传递给客户，让客户自觉自愿而又明智地做出选择。

7.4.2　从流程驱动到分析驱动的转变

在大数据时代，谁拥有数据，谁就拥有智慧。以往，企业的经营管理一直依赖于规范的流程和制度，人的主观能动性甚至被流程所僵化。依靠大数据应用，分析驱动的决策代替了直觉和常识，流程中每一个环节的智慧被激发出来，企业将变得更有洞察力，更具执

行力，也更加有智慧。

7.4.3 从私有资源到公共服务的转变

在大数据时代，谁拥有数据，谁就拥有财富。初期，大数据应用仅被用于自身洞察力的提升。随着数据成为一种资源，资源价值最大化的动力将促使领先企业更多地将公共资源转换成服务，从而发挥规模效益。数据资源公共服务化也让更多没有能力开发数据资源的中小企业也能够共享大数据的价值。